```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

/* 関数の定義 */
double func(double x);
double exact(double x);
/* 比較関数 */
int double_comp( const void *s1 , const void *s2 );
/* 最大値ノルムの計算 a[m...n] */
double vector_norm_max( double *a, int m, int n );
/* 行列の領域確保 */
double **dmatrix(int nr1, int nr2, int nl1, int nl2);
/* 行列の領域解放 */
void free_dmatrix(double **a, int nr1, int nr2, int nl1, int nl2);
/* ベクトル領域の確保 */
double *dvector(int i, int j);
/* 領域の解放 */
void free_dvector(double *a, int i);
/* 修正コレスキー分解 */
double **cholesky_decomp( double **a, int n );
/* 修正コレスキー分解を利用して連立一次方程式を解く */
double *cholesky_solve( double **a, double *b, int n );
/* 境界値問題を解く */
double *bvp(double *b, double a1, double a2, double u0,
            double un, int n, double (*f)(double) );

int main(void)
{
  int i, n;
  double *u, h;

  printf("分割数を入力してください--->");
  scanf("%d",&n);

  u = dvector( 1, n-1 );
  u = bvp( u, 0.0, 1.0, 0.0, 0.0, n, func );
  h = 1.0 / n;

  printf("求める答え u と誤差の最大値 e は次の通りです\n");
  for ( i = 1; i <= n-1; i++ ) printf("u[%d]=%f\n", i, u[i]);
  for ( i = 1; i <= n-1; i++ ) u[i] = u[i]-exact(i*h);
  printf("e=%f\n",vector_norm_max( u, 1, n-1 ));

  /* 領域の解放 */
  free_dvector( u, 1 );

  return 0;
}

/* 境界値問題を解く */
double *bvp(double *b, double a1, double a2, double u0,
            double un, int n, double (*f)(double) )

{
  double h, h2, **a;   /* 区間は [a1, a2] */
  int i, j;                    /* u0 と un は境界値 */

  h = (a2-a1)/n;  /* 刻み幅 */
  h2 = h*h;
  a = dmatrix( 1, n-1, 1, n-1 ); /* 係数行列 */

  /* 行列の作成 */
  for ( i = 2; i <= n-2; i++ )
  {
    for ( j = 1; j <= n-1; j++ )
    {
      a[i][j] = 0.0;
    }
    a[i][i] = 2.0; a[i][i+1] = -1.0; a[i][i-1] = -1.0;
  }
  for ( j = 3; j <= n-1; j++ ) a[1][j] = 0.0;
  a[1][1] = 2.0; a[1][2] = -1.0;
  for ( j = 1; j <= n-3; j++ ) a[n-1][j] = 0.0;
  a[n-1][n-2] = -1.0; a[n-1][n-1] = 2.0;

  /* 右辺ベクトルの作成 */
  for ( i = 1; i <= n-1 ; i++ ) b[i] = h2 * (*f)( a1 + h*i ) ;
  b[1] += u0; b[n-1] += un;

  /* 修正コレスキー分解 */
  a = cholesky_decomp( a, n-1 );
  /* 修正コレスキー分解を利用して連立一次方程式を解く */
  b = cholesky_solve( a, b, n-1 );

  /* 領域の解放 */
  free_dmatrix( a, 1, n-1, 1, n-1);
  return b;
}

/* 関数の定義 */
double func(double x)
{
  return( 20.0*x*x*x );
}

double exact(double x)
{
  return( x - pow(x,5.0) );
}


double **cholesky_decomp( double **a, int n )
{
  int i, j, k;
  double tmp;

  for( i = 2; i <= n; i++)
  {
    for( j = 1; j <= i-1; j++)
    {
      tmp = 0.0;
      for ( k = 1; k <= j-1; k++)
      {
        tmp += a[i][k]*a[k][k]*a[j][k];
      }
      a[i][j] = (a[i][j] - tmp) / a[j][j];
    }
      tmp = 0.0;
      for ( k = 1; k <= j-1; k++)
      {
        tmp += a[i][k]*a[i][k]*a[k][k];
      }
      a[i][i] = a[i][i] - tmp;
  }
  return a;
}

double *cholesky_solve( double **a, double *b, int n )
{
  int i, j;
  double tmp;

  /* LDy = b */
  b[1] = b[1]/a[1][1];
  for( i = 2; i <= n; i++)
  {
    tmp = 0.0;
    for( j = 1; j <= i-1; j++)
    {
      tmp += a[j][j]*a[i][j]*b[j];
    }
    b[i] = ( b[i] - tmp ) / a[i][i];
  }

  /* L^t x = y */
  for( i = n-1; i >= 1; i--)
  {
    tmp = 0.0;
    for( j = i+1; j <= n; j++)
    {
      tmp += a[j][i] * b[j];
    }
    b[i] = b[i] - tmp;
  }

  return b;
}

double **dmatrix(int nr1, int nr2, int nl1, int nl2)
{
  int i, nrow, ncol;
  double **a;

  nrow = nr2 - nr1 + 1; /* 行の数 */
  ncol = nl2 - nl1 + 1; /* 列の数 */

  /* 行の確保 */
  if ((a = malloc(nrow * sizeof(double *))) == NULL)
  {
    printf("メモリが確保できません (行列 a)\n");
    exit(1);
  }
  a = a - nr1; /* 行をずらす */

  /* 列の確保 */
  for (i = nr1; i <= nr2; i++)
    a[i] = malloc(ncol * sizeof(double));
  for (i = nr1; i <= nr2; i++)
    a[i] = a[i] - nl1; /* 列をずらす */

  return (a);
}

void free_dmatrix(double **a, int nr1, int nr2, int nl1, int nl2)
{
  int i;

  /* メモリの解放 */
  for (i = nr1; i <= nr2; i++)
    free((void *)(a[i] + nl1));
  free((void *)(a + nr1));
}

double *dvector(int i, int j) /* a[i]~a[j] の領域を確保 */
{
  double *a;

  if ((a = malloc(((j - i + 1) * sizeof(double)))) == NULL)
  {
    printf("メモリが確保できません (from dvector) \n");
    exit(1);
  }

  return (a - i);
}

void free_dvector(double *a, int i)
{
  free((void *)(a + i)); /* (void *) 型へのキャストが必要 */
}

/* 比較関数 (昇順) */
int double_comp(const void *s1, const void *s2)
{
        const double a1 = *((double *)s1); /* (double *) へキャスト */
        const double a2 = *((double *)s2); /* (double *) へキャスト */
        if (a1 < a2)
        {
                return -1;
        }
        else if (a1 == a2)
        {
                return 0;
        }
        else
        {
                return 1;
        }
}

/* 最大値ノルムの計算 a[m...n] */
double vector_norm_max(double *a, int m, int n)
{
        int i, tmp;
        tmp = n - m + 1; /* 全要素数の計算 */
        for(i = m; i <= n; i++) a[i] = fabs(a[i]);
        /* 並べ換え:先頭アドレスが m だけずれていることに注意 */
        qsort(a + m, tmp, sizeof(a[0]), double_comp);
        return a[n];
}
```