Numerische Methoden II, 170 024, und Übungen, 170 025

Course Notes, Summer Term 2017

Some material and exercises

The Laplace equation, cylindrically or spherically symmetric case

Electric and gravitational potential, waves or vibrations are just a few typical applications where

$$\Delta u = 0 \qquad \text{(Laplace equation)}$$
$$\Delta u = q \qquad \text{(Poisson equation)}$$

occur. In these equations, Δ denotes the Laplace-Operator, which in three-dimensional cartesian xyz-coordinates is

$$\Delta = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2} \tag{1}$$

The cylindrical coordinate system uses polar coordinates r and ϕ to locate a point in the $x-y$ plane and the coordinate z for the height of the point above the plane. The Laplace operator in this coordinate system is

$$\Delta = \frac{\partial^2}{\partial r^2} + \frac{1}{r}\frac{\partial}{\partial r} + \frac{1}{r^2}\frac{\partial^2}{\partial \phi^2} + \frac{\partial^2}{\partial z^2}$$

For functions depending on r (distance from z axis) only (cylindrical symmetry), this reduces to

$$\Delta = \frac{\partial^2}{\partial r^2} + \frac{1}{r}\frac{\partial}{\partial r}. \tag{2}$$

Similarly, in spherical symmetry (where r measures the distance from the orign)

$$\Delta = \frac{\partial^2}{\partial r^2} + \frac{2}{r}\frac{\partial}{\partial r}. \tag{3}$$

These differential operators (2,3) occur in the following examples, which therefore in fact are just special cases of partial differential equations, reduced to ordinary differential equations by symmetry. Many features of our simple examples are also typical for "real" Laplace- and Poisson equations.

We use examples in cylindrical symmetry here just because one-dimensional versions of the Laplace operator (1) in cartesian geometry would not include first derivatives. Solutions in that case are so simple that there is not much to learn from. (On the other hand: what's wrong with simple examples? We shall discuss also the simple cases.)

Steady-state diffusion or heat conduction

A cylindrically symmetric Laplace equation, cf. (2) governs stationary heat conduction or diffusion with axial symmetry. An example would be a cylindrical pipe, interior radius r_0, exterior radius r_1. The temperature $\theta = \theta(r)$ in the wall of the pipe follows the equation

$$\theta_{rr} + \frac{1}{r}\theta_r = 0 \quad \text{for } r_{int} < r < r_{ext}. \tag{4}$$

Dirichlet boundary conditions specify the temperature at the interior and exterior walls,

$$\begin{aligned} \theta(r_{int}) &= \theta_{int} && \text{(“inner wall temperature”),} \\ \theta(r_{ext}) &= \theta_{ext} && \text{(“outer wall temperature”),} \end{aligned}$$

(Note: there are several ways to denote derivatives of a function like $\theta(r)$.

$$\frac{d^2\theta(r)}{dr^2}, \quad \theta'', \quad \theta_{rr}$$

all stand for the same differential operator. Depending on context, one way or another may be more convenient to work with.)

For this problem, the exact solution is known. We will use it to check the accuracy of our numerical approximations.

$$\theta(r) = \frac{\theta_{int}\log(r/r_{ext}) + \theta_{ext}\log(r_{int}/r)}{\log(r_{int}/r_{ext})}.$$

Alternatively, we might prescribe the heat flow rate q at the inner boundary (A Dirichlet condition at the right and a *Neumann condition* at the left end). In that case,

$$\theta_r(r_{int}) = \alpha, \quad \theta(r_{ext}) = \theta_{ext}\ .$$

The exact solution here is

$$\theta(r) = \theta_{ext} + r_{int}\alpha\log\frac{r}{r_{ext}}.$$

For this differential equation, Neumann boundary conditions on both sides are no good choice. If the corresponding rates are equal, the solution is unique only up to an arbitrary additive constant. If the rates are not equal, no steady state can develop; thus, no solution exists! This simple example demonstrates that boundary value problems do not necessarily have a solution. (in contrast to initial value problems, where existence of a solution is guaranteed in the vicinity of the starting point for a wide class of functions)

Dirichlet and Neumann conditions are also called *boundary conditions of the first and second kind*, respectively. There are third-kind boundary

conditions as well (some attach to them the names Robin or Sturm, or call them radiation conditions), of the form $c_1 p(r) + c_2 p'(r) = \alpha$. Also *nonlinear* boundary conditions may occur, like $c_1 p(r)^4 + c_2 p'(r) = \alpha$, which would model thermal radiation according to Stefan-Boltzmann's law.

Cooling of cylindrical objects

To find solutions of the time-dependent heat equation in cylindrical symmetry,

$$\theta_t = a\left(\theta_{rr} + \frac{1}{r}\theta_r\right)$$

we set $\theta(r,t) = e^{-\lambda^2 at} u(r)$, which reduces the problem to an ordinary differential equation for $u(r)$,

$$u_{rr} + \frac{1}{r}u_r + \lambda^2 u = 0, \quad 0 < r < R \tag{5}$$

which is (after multiplying by r^2 and a transformation of variables $r \to \lambda r$) a form of Bessel's differential equation. Symmetry dictates a Neumann boundary condition $u_r = 0$ at the center $r = 0$. For an outer Dirichlet boundary condition $u = 0$ at $r = R$, it turns out that *nontrivial* solutions exist for certain values of λ only (the trivial solution would be $u(r) = 0$ for all r).

The solution involves J_0, the Bessel function of order zero of the first kind,

$$u(r) = A J_0(\lambda r)$$

where A is an arbitrary constant (remember: the problem is a homogeneous one; any multiple of a solution is also a solution). The outer boundary condition then requires λ to be a root of

$$J_0(\lambda R) = 0$$

Steady one-dimensional convection-diffusion equation

Let $c = c(x)$ denote the concentration of some quantity in a flow with velocity w and a coefficient of diffusion D. Then

$$\begin{aligned} Dc'' - wc' &= 0 \quad \text{for} \quad 0 < x < L, \\ c &= c_0 \text{ at } x = 0, \\ c &= c_L \text{ at } x = L, \end{aligned}$$

models a steady-state situation where convective transport is balanced by diffusion. This problem has the exact solution

$$c(r) = c_0 + \frac{e^{\mathrm{Pe}\frac{x}{L}} - 1}{e^{\mathrm{Pe}} - 1}(c_L - c_0).$$

Here Pe is the Peclet number, $\mathrm{Pe} = wL/D$.

Ferzinger and Perić comment on this problem:

> "Because it is so simple, this problem is often used as a test of numerical methods, including both discretization and solution schemes. [...] There are few actual flows in which this balance plays an important role. Normally, convection is balanced by either a pressure gradient or diffusion in the direction normal to the flow. [...] Indeed, use of this problem as a test case has probably produced more poor choices of method than any other in the field."

Discretization on an Equidistant Grid, Dirichlet Conditions

We discuss the example

$$\begin{aligned} \theta'' + \frac{1}{r}\theta' &= 0 \quad \text{for } \theta = \theta(r), \quad 0 < r_{int} < r < r_{ext} \\ \theta(r_{int}) &= \theta_{int}, \\ \theta(r_{ext}) &= \theta_{ext}. \end{aligned}$$

The first step in the application of finite difference methods is the selection of a set of mesh points in the interval (r_{int}, r_{ext}). Let us simply take n equidistant radius points r_j,

$$r_j = r_{int} + jh \quad \text{with } \textit{mesh size} \quad h = \frac{r_{ext} - r_{int}}{n+1}, \quad j = 1, \ldots, n$$

Boundary points are $r_0 = r_{int}$ and $r_{n+1} = r_{ext}$. Next, we represent the differential equation by difference equations involving values of the unknown pressure at the mesh points. A straightforward way to do so is to replace the derivatives by appropriate difference quotients.

Standard approximations are the difference formulae

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h}$$

$$f''(x) \approx \frac{f(x+h) - 2f(x) + f(x-h)}{h^2}$$

Notation: For the value $\theta(r)$ of the unknown temperature at the mesh point r_j we write θ_j. At mesh point $r_{j+1} = r_j + h$, it is θ_{j+1}, similarly θ_{j-1} at r_{j-1}. This way we may write

$$\theta'_j \approx \frac{1}{2h}\left(\theta_{j+1} - \theta_{j-1}\right)$$

$$\theta_j'' \approx \frac{1}{h^2}\left(\theta_{j+1} - 2\theta_j + \theta_{j-1}\right)$$

Substitution leads to difference equations, one per interior mesh point. The temperature values θ_j at all interior mesh points fulfill approximately

$$-\left(1 - \frac{h}{2r_j}\right)\theta_{j-1} + 2\theta_j - \left(1 + \frac{h}{2r_j}\right)\theta_{j+1} = 0 \quad \text{for} \quad j = 1, \ldots, n. \tag{6}$$

This is a system of n linear algebraic equations. In matrix form,

$$A \cdot \mathbf{x} = \mathbf{b} \quad \text{with} \tag{7}$$

$$A = \begin{bmatrix} 2 & -(1+\frac{h}{2r_1}) & & & & & \\ -(1-\frac{h}{2r_2}) & 2 & -(1+\frac{h}{2r_2}) & & & & \\ & -(1-\frac{h}{2r_3}) & 2 & -(1+\frac{h}{2r_3}) & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & -(1-\frac{h}{2r_{n-1}}) & 2 & -(1+\frac{h}{2r_{n-1}}) & \\ & & & & -(1-\frac{h}{2r_n}) & 2 & \end{bmatrix},$$

$$\vec{x} = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \\ \vdots \\ \\ \theta_n \end{bmatrix}, \quad \vec{b} = \begin{bmatrix} (1-\frac{h}{2r_1})\theta_{int} \\ 0 \\ 0 \\ \\ \vdots \\ \\ (1+\frac{h}{2r_n})\theta_{ext} \end{bmatrix}.$$

Note that this matrix is not symmetric, but diagonally dominant as long as $h < 2r_1$.

Exercises

Exercise 1 Solve the dirichlet problem (4) for steady-state radial flow for two cases.

1. Strong influence of radial geometry:

$$r_{int} = 0.1;\ r_{ext} = 10;\ \theta_{int} = -1;\ \theta_{ext} = 0.$$

2. moderate influence of radial geometry:

$$r_{int} = 1;\ r_{ext} = 10;\ \theta_{int} = -1;\ \theta_{ext} = 0.$$

To evaluate the results, plot

1. The error as a function of r for some values of n, say $n = 10, 100, 1000$.
2. The maximum error as a function of n for a range of n as large as computer power allows. Show the error in a log-log-Plot and estimate the asymptotic behavior as $n \to \infty$. An important quantity is the slope of the curve.

What we shall find out:

- For large n the maximum error decreases proportional to n^{-2}. This quadratic convergence, however, does not become apparent unless the grid size h is comparable to the interior radius r_{int}.
- The memory required is proportional to n. Can you specify the memory requirements of your program more precisely?

Exercise 2 Derive a difference approximation for the convection-diffusion equation and set up the system of linear equations. Is the matrix symmetric? diagonally dominant? Try to solve the simple case $D = 1, L = 1, w = 1, c_0 = 1, c_L = 0$ for various h, and compare results with the exact solution. Change $w = 100$ and try again. Evaluate your results with similar plots as in Exercise 1.

Exercise 3 The matrix (7) in our radial flow example is not symmetric. Show that by multiplying the i-th equation by r_i a symmetric matrix results. Systems with symmetric matrices are theoretically and numerically in many ways easier to handle than non-symmetric ones. Implement the symmetric version and check with the original version.

Exercise 4 Test the sensitivity of the system (7) to small perturbations in the right-hand side. In our MATLAB code, insert

```
% original solution
p = A\b;

%solution with perturbed right-hand side
% try different perturbations
err=rand(n,1)*0.00001;

%err=(rand(n,1)-0.5)*0.00001;

b = b+err;
perr = A\b;
```

The relative error in b is norm(err)/norm(b), while the relative error in the solution is norm(p-perr)/norm(p).

How much larger is the relative error in the solution in relation to the relative error of the right-hand side? Calculate this quotient (a measure of the "sensitivity" of the linear system) for different values of n and estimate the behavior for large n. Show a loglog-plot.

In MATLAB, c = cond(A) returns the 2-norm condition number, the ratio of the largest singular value of A to the smallest. The condition number of a matrix measures the sensitivity of the solution of a system of linear equations to errors in the data. It gives an indication of the accuracy of the results from matrix inversion and the linear equation solution. It also is an upper bound on the "sensitivity", as calculated before. Values of cond(A) near 1 indicate a well-conditioned matrix: errors in the data generate errors of the same magnitude in the solution. Add a plot of the condition number depending on n to your loglog-Plot.

Is the condition number a quite sharp or a rather pessimistic estimate on the sensitivity in this example?

Discretization on an Equidistant Grid, Mixed Boundary Conditions

Just a few modifications in our first program are necessary to implement a Neumann condition at the inner radius and a Dirichlet condition at the outer boundary. We are now going to solve our second example of flow in porous media, the problem

$$\begin{aligned} p'' + \frac{1}{r}p' &= 0 \quad \text{for } p = p(r), \quad r_w < r < r_e \\ p'(r_w) &= \alpha, \\ p(r_e) &= p_e. \end{aligned} \tag{8}$$

We may use the same grid and the same difference formula as before, but there is one problem: For the first interior gridpoint r_1, the difference approximation is

$$-\left(1 - \frac{h}{2r_1}\right) p_0 + 2p_1 - \left(1 + \frac{h}{2r_1}\right) p_2 = 0. \tag{9}$$

However, the boundary condition does not give us any *value* for p_0, it just specifies the *derivative* there. We can exploit this information by using a one-sided approximation of the first derivative,

$$f'(x) = \frac{1}{h}\Big(-f(x) + f(x+h)\Big) - h\frac{f''(x)}{2} + \mathcal{O}(h^2). \tag{10}$$

In our case, we set

$$\frac{p_1 - p_0}{h} = \alpha, \quad \text{or } p_0 = p_1 - h\alpha. \tag{11}$$

Inserting for p_0 in Equation (9), we obtain a difference equation for the first interior gridpoint at a Neumann boundary,

$$\left(1 + \frac{h}{2r_1}\right) p_1 - \left(1 + \frac{h}{2r_1}\right) p_2 = -\alpha h \left(1 - \frac{h}{2r_1}\right). \tag{12}$$

There is an inconsistency with using this approximation along with the rest of the difference equations. Because of the one-sided approximation (10), 12 is a fist-order approximation only. At all other points, the difference equations are second-order approximations to the original differential equation. The first-order approximation of the Neumann condition does indeed contaminate our final results in the whole region. In general, the order of approximation of a problem is the lowest order used in any parts of the problem. Sometimes, however, a bad approximation at just a few points will not affect the overall accuracy.

To get a second-order accurate approximation at r_0, we can replace the second derivative in 10 using the differential equation. In our case,

$$\begin{aligned} p'(r_0) &= \frac{p_1 - p_0}{h} - \frac{h}{2} p''(r_0) + \mathcal{O}(h^2) \\ p''(r_0) &= -\frac{1}{r} p'(r_0) \quad \text{from the differential equation} \\ &= -\frac{\alpha}{r} \qquad \text{from the boundary condition} \end{aligned}$$

From these equations follows

$$p_0 = p_1 - h\alpha + \frac{h^2 \alpha}{2r} \tag{13}$$

More commonly, however, Neumann boundaries are treated by modifying the grid. Instead of

$$r_0 = r_w, \quad r_1 = r_w + h,$$

the modified grid at a Neumann boundary has

$$r_0 = r_w - \frac{h}{2}, \quad r_1 = r_w + \frac{h}{2}$$

All gridpoints are given by

$$r_i = r_w + (i - 1/2)h \quad \text{with} \quad h = \frac{r_e - r_w}{n + 1/2}, \quad i = 1, \ldots, n$$

The point r_0 is now outside of the region where our problem is defined. (It is called a ghost point.)

As a result of this modification, the approximation $p_0 = p_1 - \alpha h$ is of second order. However, this scheme does not compute the pressure p_{wf} directly, since there is no gridpoint at r_w. We may set $p_{wf} = (p_0 + p_1)/2 + \mathcal{O}(h^2)$.

Exercises

Exercise 5 Solve the mixed-type problem (8) and test the three ways to implement Neumann boundary conditions. Data:

$$r_w = 0.1;\ r_e = 10;\ p'(r_w) = \alpha = 1;\ p_e = 0.$$

To evaluate the results, plot the maximum error as a function of n in a log-log-Plot and estimate the asymptotic behavior as $n \to \infty$. The difference between first-order and second-order accuracy should become clearly apparent.

Exercise 6 Solve for $r_w = 0,1$ and $r_e = 10$ the inhomogeneos differential equation with homogeneous mixed-type boundary conditions (quasi-steady-state flow),

$$\begin{aligned} p'' + \frac{1}{r}p' &= 1 \quad \text{for } r_w < r < r_e, \\ p'(r_e) &= 0, \\ p(r_w) &= 0. \end{aligned}$$

Evaluate the accuracy of your solution as in Exercise 1. (If you have coded this exercise, the modifications in the code should be rather marginal!)

Exercise 7 Discretize the differential equation (5). Assume simple values for a and λ. What happens to the right-hand side of the system? What solution gives MATLAB's `A\b` ? Any ideas how to proceed in this case?

Deriving Difference Approximations for Derivatives

Consider a smooth function $f(x)$. ("Smooth" here means that as many derivatives of f exixt as we may need.)

We will illustrate, by way of example, a method to find approximations to derivatives. It is sometimes called the *method of undetermined coefficients*.

Example: Derive an approximation for f'' at gridpoint x_0 using values at gridpoints x_{-1}, x_0 and x_1 of an equidistant grid with spacing h.

Expand f_{-1} and f_1 in Taylor series around x_0

$$\begin{aligned} f_{-1} &= f_0 - hf_0' + \frac{h^2}{2}f_0'' - \frac{h^3}{6}f_0''' + \frac{h^4}{24}f_0^{\mathrm{IV}} + \mathcal{O}(h^5) \\ f_1 &= f_0 + hf_0' + \frac{h^2}{2}f_0'' + \frac{h^3}{6}f_0''' + \frac{h^4}{24}f_0^{\mathrm{IV}} + \mathcal{O}(h^5) \end{aligned}$$

We now seek a linear combination of the three values f_{-1}, f_0 and f_1 approximating f_0'' in the form

$$f_0'' \approx af_{-1} + bf_0 + cf_1.$$

To determine the coefficients a, b and c, we insert the Taylor series and collect terms in $f_0, f_0', \ldots$. We find

$$\begin{aligned} f_0'' \approx\ & (a+b+c)f_0 + h(-a+c)f_0' + \frac{h^2}{2}(a+c)f_0'' + \\ & \frac{h^3}{6}(-a+c)f_0''' + \frac{h^4}{24}(a+c)f_0^{\mathrm{IV}} + \mathcal{O}(h^5) \end{aligned}$$

Since we have three degrees of freedom (three undetermined coefficients), we can plan on setting the coefficients of f_0 and f_0' equal to zero and the coefficient of f_0'' equal to one. We are left with the following system of equations.

$$\begin{aligned} a+b+c &= 0 \\ -a+c &= 0 \\ a+c &= \frac{2}{h^2} \end{aligned}$$

Solving the above system of equations gives

$$a = \frac{1}{h^2}, \quad b = -\frac{2}{h^2}, \quad c = \frac{1}{h^2}.$$

As an additional bonus, this solution makes the coefficient of f_0''' equal to zero too. The coefficient of f_0^{IV} turns out to be $h^2/12$. Hence, we get the approximation

$$\frac{1}{h^2}f_{-1} - \frac{2}{h^2}f_0 + \frac{1}{h^2}f_1 = f_0'' + \frac{h^2}{12}f_0^{\mathrm{IV}} + \mathcal{O}(h^3).$$

This is the well-known centered difference approximation that we have used already.

As an additional exercise, you may derive a five-point centered approximation to f_0'''. The system of equations for five coefficients a, b, c, d and e in this case will be

$$\begin{aligned} a+b+c+d+e &= 0 \\ -2a-b+d+2e &= 0 \\ 4a+b+d+4e &= 0 \\ -8a-b+d+8e &= 6/h^3 \\ 16a+b+d+16e &= 0 \end{aligned}$$

(Using an algebraic manipulator such as Maple or Mathematica makes solving such systems much easier!) The solution is

$$a = -\frac{1}{2h^3}, \quad b = \frac{1}{h^3}, \quad c = 0, \quad d = -\frac{1}{h^3}, \quad e = \frac{1}{2h^3}.$$

An alternative method: find an interpolating polynomial $p(x)$ for the data points (x_i, f_i). Differentiate the polynomial with respect to x and evaluate at x_0.

If you do this by hand, you should use the *Lagrange interpolation formula*:

The polynom $p(x)$ that interpolates the $n+1$ pairs of values

$$(x_0, f_0),\ (x_1, f_1), \ldots, (x_n, f_n)$$

is given by

$$p(x) = f_0 L_0(x) + f_1 L_1(x) + \cdots + f_n L_n(x)\ ,$$

where

$$L_k(x) = \frac{(x-x_0)(x-x_1)\cdots(x-x_{k-1})(x-x_{k+1})\cdots(x-x_n)}{(x_k-x_0)(x_k-x_1)\cdots(x_k-x_{k-1})(x_k-x_{k+1})\cdots(x_k-x_n)}$$

for each $k = 0, 1, \ldots, n$.

In Mathematica, this method conveniently finds difference approximations for higher derivatives.

The commands

```
data={{-2h,f[-2h]},{-h,f[-h]},{0,f[0]},{h,f[h]},{2h,f[2h]}};
poly=InterpolatingPolynomial[data,x];
appr=D[poly,{x,3}]/.x->0 //Together
```

find an interpolating polynomial and evaluate its third derivative at $x = 0$. Mathematica gives

$$\text{appr} = \frac{-f(-2\,h) + 2\,f(-h) - 2\,f(h) + f(2\,h)}{2\,h^3}.$$

We may find the truncation error by the command

```
Series[appr,{h,0,3}]
```

which produces the output

$$f^{(3)}(0) + \frac{f^{(5)}(0)\,h^2}{4} + \mathrm{O}(h)^4.$$

Since many tables provide difference approximations for f', f'', ..., it is normally not necessary to derive basic formulae. However, for irregular grids, complicated equations or some specific boundary conditions, these methods are of great value.

Non-Equidistant Grid

Write

$$f^- = f(x - h^-), \quad f^0 = f(x), \quad f^+ = f(x + h^+),$$
$$h^- = r_i - r_{i-1}, \quad h^+ = r_{i+1} - r_i.$$

Then

$$f'(x) = \frac{(f^+ - f^0)\frac{h^-}{h^+} + (f^0 - f^-)\frac{h^+}{h^-}}{h^- + h^+} - \frac{h^+ h^-}{6} f'''$$

$$f''(x) = \frac{2}{h^- + h^+}\left(\frac{f^+ - f^0}{h^+} - \frac{f^0 - f^-}{h^-}\right) - \frac{h^+ - h^-}{3} f''' - \frac{(h^+)^2 - h^+ h^- + (h^-)^2}{12} f^{\mathrm{IV}}$$

Note that in the second approximation, for $h^+ \neq h^-$ now there is a low-order truncation error term involving f'''. Equidistant grids usually provide approximations with higher-order truncation errors. So you have to balance the gain by finer meshsize against possibly higher truncation error.

Linear solvers

The Thomas Algorithm for tridiagonal Systems

The so-called Thomas Algorithm is just a form of elimination for solving tridiagonal systems of linear equations. Llewellyn H. Thomas used it around 1950

to solve elliptic partial differential equations. The attribution to Thomas seems to be more common in some engineering disciplines than it is in numerical analysis.

Let A be a tridiagonal matrix and b the right-hand side of a system $Ax = b$.

$$A = \begin{bmatrix} d_1 & e_1 & & & & \\ c_2 & d_2 & e_2 & & & \\ & c_3 & d_3 & e_3 & & \\ & & \ddots & \ddots & \ddots & \\ & & & c_{n-1} & d_{n-1} & e_{n-1} \\ & & & & c_n & d_n \end{bmatrix}, \quad b = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_{n-1} \\ b_n \end{bmatrix}.$$

Transform it to upper triangular form, by Gaussian elimination. Use backsubstitution to solve the resulting equivalent system $Ax' = b'$

$$A' = \begin{bmatrix} 1 & e'_1 & & & & \\ & 1 & e'_2 & & & \\ & & 1 & e'_3 & & \\ & & & \ddots & \ddots & \\ & & & & 1 & e'_{n-1} \\ & & & & & 1 \end{bmatrix}, \quad b' = \begin{bmatrix} b'_1 \\ b'_2 \\ b'_3 \\ \vdots \\ b'_{n-1} \\ b'_n \end{bmatrix}.$$

The expressions for the entries in A' and b' follow from a step-by-step application of the usual elimination procedure.

$$e'_1 = \frac{e_1}{d_1}, \quad b'_1 = \frac{b_1}{d_1};$$

for $i = 2, \ldots, n$:

$$e'_i = \frac{e_i}{d_i - c_i e'_{i-1}}, \quad b'_i = \frac{b_i - c_i b'_{i-1}}{d_i - c_i e'_{i-1}};$$

Backsubstitution: $x_n = b'_n$, und f"ur $i = n-1, \ldots, 1$:

$$x_i = b'_i - e'_i x_{i+1}.$$

This algorithm is stable if

$$\begin{cases} d_i > 0, & i = 1, 2, \ldots, n, \\ d_1 > |e_1|, & \\ d_i \geq |c_i| + |e_i| \text{ und } c_i \neq 0, e_i \neq 0, & i = 2, \ldots, n-1, \\ d_n \geq |c_n|. & \end{cases}$$

Essentially these conditions require A to be (weakly) diagonally dominant. They are sufficient but not necessary.