Lecture # 29
Two-Point Boundary Value Problems and Other Matters

First things first.
There is a matrix $A$ (actually many) such that

$$\rho(A) < \|A\|$$

for any induced norm $\| \cdot \|$. One such matrix is

$$A = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}. $$

Note that

$$\rho(A) = 0,$$

but since $A \neq 0$, for any norm

$$\|A\| > 0.$$  

One common form for the two-point boundary value problem is

\begin{align*}
y''(x) &= f(x, y, y') \quad (1) \\
y(a) &= \beta_1, \quad y(b) = \beta_2. \quad (2)
\end{align*}

Using our usually first order system transformation this is

$$y_1 = y, \quad y_2 = y'.$$

and becomes

\begin{align*}
y_1'(x) &= y_2(x) \\
y_2'(x) &= f(x, y_1, y_2) \\
y_1(a) &= \beta_1, \quad y_1(b) = \beta_2.
\end{align*}

Notice that there are no conditions on $y_2(\cdot)$ at all. Some existence conditions are discussed in your book. Below is one example.
Example 1

\[ y''(x) = y \]
\[ y(0) = 27, \quad y(\pi/2) = -33 \]

All of the solutions of the first equation are of the form

\[ y(x) = \alpha_1 \sin x + \alpha_2 \cos x. \]

Using the boundary conditions

\[ y(0) = 27 = \alpha_2, \quad y(\pi/2) = -33 = \alpha_1. \]

Thus

\[ y(x) = -33 \sin x + 27 \cos x. \]

To simplify the discussion (this is just a taste, after all), we consider the linear two-point boundary value problem

\[ y'(x) = A(x)y(x) + \phi(x), \quad a < x < b \]
\[ B_a y(a) + B_b y(b) = \beta \]

(3)

To solve this correctly, we have solve a lot of initial value problems.

Last time, we did simple shooting. First, we solve for a matrix function \( Y(x) \in \mathbb{R}^{n \times n} \) such that

\[ Y'(x) = A(x)Y(x) \]
\[ Y(a) = I \]

The matrix \( Y(x) \) is the general solution to this ordinary differential equation. Then we solve for a particular solution to

\[ v'(x) = A(x)v(x) + \phi(x) \]
\[ v(a) = 0 \]

Sometimes we choose \( v(a) = \alpha \), but zero initial conditions are usually convenient.

All solutions to this ordinary differential equation have the form

\[ y(x) = Y(x)s + v(x) \]
We note that
\[ y(a) = s, \]
so we are, in effect, choosing the initial conditions. To find \( s \), we plug \( y(x) \) into the initial conditions, thus
\[ B_a y(a) + B_b y(b) = \beta \]
so that
\[ B_a s + B_b (Y(b)s + v(b)) = \beta. \]
Thus,
\[ Ms = \hat{\beta} \]
where
\[ M = B_a + B_b Y(b) \]
and
\[ \hat{\beta} = \beta - B_b v(b). \]

Since the \((n + 1)\) ordinary differential equations are solved on different meshes, we have to solve the \((n + 2)^{nd}\) equation
\[ y'(x) = A(x)y(x) + \phi(x) \]
\[ y(a) = s \]
to get the solution.

Simple shooting has some difficulties. The computed \( y \) may not satisfy the boundary conditions that well and the matrix \( M \) can be very badly conditioned.

Instead, we use multiple shooting. We still do about the work of solving \((n + 1)\) initial value problems. We pick mesh points
\[ a = x_0 < x_1 < \ldots < x_N < x_{N+1} = b. \]
For each interval \([x_j, x_{j+1}], j = 0, 1, \ldots, N\) we solve
\[ Y'_j(x) = A(x)Y_j(x) \]
\[ Y_j(x_j) = F_j \]
Usually, \( F_j \) is chosen to be \( I_n \). We then solve
\[ v_j(x) = A(x)v_j(x) + \phi(x) \]
\[ v_j(x_j) = \alpha_j \]
Again, $\alpha_j$ is often chosen to be 0, but not always.

Then

$$y(x_j) = Y_j(x_j)s_j + v_j(x_j) = F_j s_j + \alpha_j, \quad j = 0, \ldots, N$$

and

$$y(x_{N+1}) = Y_N(x_{N+1})s_N + v_N(x_{N+1}).$$

We exploit continuity, note that

$$y(x_{j+1}) = Y_j(x_{j+1})s_j + v_j(x_{j+1}) = F_{j+1} s_{j+1} + \alpha_{j+1}$$

so that

$$Y_j(x_{j+1})s_j - F_{j+1} s_{j+1} = \alpha_{j+1} - v_j(x_{j+1}), \quad j = 0, \ldots, N - 1.$$

We then add the boundary conditions. Namely,

$$B_a y(x_0) + B_b y(x_N) = \beta$$

which leads to

$$B_a (F_0 s_0 + \alpha_0) + B_b (Y_N(x_{N+1})s_N + v_N(x_{N+1}) = \beta.$$

Some rearranging yields

$$B_a F_0 s_0 + B_b Y_N(x_{N+1})s_N = \beta - B_a \alpha_0 - B_b v_N(x_{N+1}).$$

Thus, we can solve for

$$s = (s_0, \ldots, s_N)^T.$$

This leads to the equation

$$Ms = \hat{\beta},$$

where

$$M = \begin{pmatrix}
B_a F_0 & \cdots & \cdots & \cdots & \cdots & B_b Y_N(x_{N+1}) \\
Y_0(x_1) & -F_1 & \cdots & \cdots & \cdots & \cdots \\
0 & Y_1(x_2) & -F_2 & \cdots & \cdots & \cdots \\
0 & 0 & Y_2(x_3) & -F_3 & \cdots & \cdots \\
\cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
\cdots & \cdots & \cdots & \cdots & Y_{N-1}(x_N) & -F_N
\end{pmatrix}$$
and

\[
\hat{\beta} = \begin{pmatrix}
\beta - B_a \alpha_0 - B_b v_N(x_{N+1}) \\
\alpha_1 - v_0(x_1) \\
\alpha_2 - v_1(x_2) \\
\alpha_3 - v_2(x_3) \\
\vdots \\
\alpha_N - v_{N-1}(x_N)
\end{pmatrix}
\]

\[
= \begin{pmatrix}
\hat{\beta}_0 \\
\hat{\beta}_2 \\
\hat{\beta}_3 \\
\vdots \\
\hat{\beta}_N
\end{pmatrix}
\]

There is a very straightforward algorithm to solve this linear system. We will compute its PLU decomposition (by Gaussian elimination).

\[M = LR.\]

The matrix is already in block bidiagonal form except for the last block row. It is somewhat large, \(n(N+1) \times n(N+1)\), but it has only two \(n \times n\) blocks per block row, so the storage is \(2n^2(N+1)\). You do not need any special data structures for this matrix either.

We let \(L\) be the result of be the product of \(N\) PLU factorizations.

\[L = L_1 \cdots L_N\]

described below. For instances, we construct \(P_1\) and \(L_1\) from \(\tilde{P}_1\) and \(\tilde{L}_1\) such that

\[
\begin{pmatrix}
1 & 2 & N+1 & rhs \\
1 & 2 & N+1 & rhs
\end{pmatrix}
\begin{pmatrix}
B_a F_0 & 0 & B_b v_N(x_{N+1}) & \hat{\beta}_0 \\
Y_0(x_1) & -F_1 & 0 & \hat{\beta}_1
\end{pmatrix}
= \tilde{P}_1 \tilde{L}_1
\begin{pmatrix}
1 & 2 & N+1 & rhs \\
1 & 2 & N+1 & rhs
\end{pmatrix}
\begin{pmatrix}
R_{11} & R_{32} & R_{1, N+1} & \tilde{\beta}_0 \\
0 & \tilde{R}_{22} & R_{2, N+1} & \tilde{\beta}_1
\end{pmatrix}
\]

and let

\[L_1 = \begin{pmatrix}
\tilde{P}_1 \tilde{L}_1 & 0 \\
0 & I_{(N-1)n}
\end{pmatrix}.
\]
Then for $k = 2, \ldots, N - 1$ we have

$$
\begin{pmatrix}
  k & k + 1 & N + 1 & \text{rhs} \\
  k + 1 & 0 & \tilde{R}_{k,N+1} & \tilde{\beta}_{k-1} \\
  Y_k(x_{k+1}) & -F_{k+1} & 0 & \tilde{\beta}_k
\end{pmatrix} = \tilde{P}_k \tilde{L}_k
\begin{pmatrix}
  k & k + 1 & N + 1 & \text{rhs} \\
  k + 1 & 0 & \tilde{R}_{k+1,N+1} & \tilde{\beta}_{k-1} \\
  0 & \tilde{R}_{k+1,k+1} & 0 & \tilde{\beta}_k
\end{pmatrix}
$$

Then

$$
L_k = \begin{pmatrix}
  I_{(k-1)n} & 0 & 0 \\
  0 & \tilde{P}_k \tilde{L}_k & 0 \\
  0 & 0 & I_{(N-k)n}
\end{pmatrix}
$$

For the last block row, we have

$$
\begin{pmatrix}
  N & N + 1 & \text{rhs} \\
  k + 1 & 0 & \tilde{R}_{N+1,N+1} & \tilde{\beta}_{N-1} \\
  Y_{N-1}(x_N) & -F_N & 0 & \tilde{\beta}_N
\end{pmatrix} = \tilde{P}_N \tilde{L}_N
\begin{pmatrix}
  N & N + 1 & \text{rhs} \\
  k + 1 & 0 & \tilde{R}_{N+1,N+1} & \tilde{\beta}_{N-1} \\
  0 & \tilde{R}_{N+1,N+1} & 0 & \tilde{\beta}_N
\end{pmatrix}
$$

Then

$$
L_N = \begin{pmatrix}
  I_{(N-1)n} & 0 \\
  0 & \tilde{P}_N \tilde{L}_N
\end{pmatrix}
$$

Thus we recover $s$ from

$$Rs = \tilde{\beta}
$$

where

$$R = \begin{pmatrix}
  R_{11} & R_{12} & 0 & 0 & R_{1,N+1} \\
  0 & R_{22} & R_{23} & 0 & R_{2,N+1} \\
  0 & 0 & R_{33} & R_{34} & R_{3,N+1} \\
  \cdots & \cdots & \cdots & \cdots & \cdots \\
  \cdots & \cdots & \cdots & R_{N,N} & R_{N,N+1} \\
  \cdots & \cdots & \cdots & \cdots & R_{N+1,N+1}
\end{pmatrix}
$$

and

$$\tilde{\beta} = \begin{pmatrix}
  \tilde{\beta}_1 \\
  \tilde{\beta}_2 \\
  \tilde{\beta}_3 \\
  \vdots \\
  \tilde{\beta}_{N+1}
\end{pmatrix}$$
We then use back substitution to get

\[
\begin{align*}
R_{N+1,N+1}s_N & = \tilde{\beta}_{N+1} \\
R_{NN}s_{N-1} & = \tilde{\beta}_N - R_{N,N+1}s_N \\
R_{kk}s_{k-1} & = \tilde{\beta}_k - R_{k,k+1}s_k - R_{k,N+1}s_N, \quad k = N - 1, \ldots, 1.
\end{align*}
\]

Each Q–R decomposition is \(O(n^3)\) operations, so altogether this requires \(O(Nn^3)\) operations.