§7. Shooting method for Boundary Value Problems.

(a) **Introduction**: initial value problems (IVP's) vs. boundary value problems (BVP's). Latter much more difficult; can't march with RK4.

- Concrete example: \[ \begin{align*}
  y'_1 &= y_2 \\
  y'_2 &= \frac{2y_1}{x^2 + 1}
\end{align*} \] \[ y_1(0) = 1, \quad y_2(0) = 0. \]

- Solution: \( y_1(x) = x^2 + 1, \quad y_2(x) = 2x \)
- Can be solved by RK4, maybe with variable step size.

**What if BVP with**: \( \begin{align*}
  y_1(0) &= 1 \\
  y_2(1) &= 2
\end{align*} \)

**Strategy**: (a) Recast into IVP by guessing the missing initial condition, say \( y_2(0) = 0.2 \).
(b) March by RK4 to the end, and see if \( y_2(1) \) matches the boundary value 2.
(c) If not, adjust guess of \( y_2(0) \) to match \( y_2(1) \) to target. But how?

- Trial and error?
- **Bisection ? Now looks like root-finding!**
- **Newton-Raphson method, with numerical differentiation**: "Discrepancy" \( F = y_2(1) - 2 \) is a function of guessed \( y_2(0) = x \). Then find \( x \) so \( F(x) = 0 \). But functional form not explicitly given \( \Rightarrow \) Numerical differentiation for \( F(x) \).

General form: \[ \begin{align*}
  \frac{dy_i(x)}{dx} &= g_i(x, y_1, \ldots, y_N), \quad i = 1, 2, \ldots, N \\
  y_j(x_1) &= y_{j1}, \quad j = 1, \ldots, n_1 \\
  y_k(x_2) &= y_{k2}, \quad k = n_1 + 1, \ldots, n_2 \\
  (n_2 = N - n_1)
\end{align*} \] over \([x_1, x_2] \).
(b) Shooting to the other boundary \([x_1, x_2]\).

(i) Expand boundary condition at \(x_1\) to form complete I.V.P.
\[y_k(x_1) = V_k, \quad k = n+1, \ldots, N.\]

(ii) March to \(x_2\) to get \(y_i(x_2), i=1, \ldots, N\)

(iii) Formulate discrepancy vector:
\[F_k = y_k(x_2) - y_{k_2}, \quad k = n+1, \ldots, N.\]

as implicitly dependent on \(V_k\) values.

(iv) Newton's method for root-finding:
\[F_k(V_k) = 0;\]

\[J \cdot \delta V = -F, \quad n_1 \times n_2 \text{ system.}\]

\[J_{ij} = \frac{\partial F_i}{\partial V_j}, \quad i,j = 1, 2, \ldots, n_2.\]

(shifted from \(n+1, \ldots, N\))

\[\frac{\partial F_i}{\partial V_j} = \frac{F_i(V, \ldots, y_i + \delta y_j, \ldots, V_{n_2}) - F_i(V, \ldots, y_i, \ldots, V_{n_2})}{\delta V_j}\]

Now we can draw the flow-chart of the shooting algorithm.

* Could guess missing BC at \(x_2\), and march back toward \(x_1\). Depends on which end has fewer "missing BC's."

Flow chart for shooting scheme:

* Module A:
  
  - Guess \( V_i \)
  - RK4, e.g.
  - Advance to other end or middle
  - Construct \( F \)
    
  * Newton-Raphson:
    
    \[ F \rightarrow E(V_i + \delta V_i) \]
    
    - \( F \)
    - Iterative \( V + \delta V \)
    - \( \delta V = -J^{-1}F \)
    - How large is \( |\delta V| \)?
      
      \[ |\delta V| < \text{tol} \]?
      
      - Yes
      - No
      
      - Done
      
      - Take solution

If \(|F| < \text{tol}, tolerate solution already! But won't be so lucky!
(c) More general expressions of BC's.

(i) BC's at \( x_1, x_2 \) needn't be neatly divided between two groups of \( y_j, j=1, \ldots, \eta_1, \eta_1+1, \ldots, \eta_1+\eta_2 \). For example:

\[
\begin{align*}
\eta_1 &= a \\
\eta_2 &= b \\
\text{Guess} \quad \eta_3 &= V_1 \\
\eta_4 &= V_2
\end{align*}
\]

\[
\begin{align*}
F_1 &= y_j(x_2) - c \\
F_2 &= y_k(x_2) - d
\end{align*}
\]

\[
\begin{pmatrix} F_1 \\ F_2 \end{pmatrix} \approx \frac{1}{2} \times 2 \text{ Newton's}
\]

(ii) BC's may be algebraic expressions. Maybe more convenient to guess values of these instead of \( y_j \):

\[
\begin{align*}
y_1 + \sqrt{y_2^2 + y_3^2} &= 2 \\
y_4 - \sin(\sqrt{y_2^2 + y_3^2}) &= 1
\end{align*}
\]

* If guess \( y_i = V_1 \)

\[
\begin{align*}
y_2 &= V_1 \\
y_3 &= V_2
\end{align*}
\]

\[
\begin{align*}
\text{Hand to compute } y_3, y_4.
\end{align*}
\]

* Guess \( \sqrt{y_2^2 + y_3^2} = V_2 \)

\[
\begin{align*}
y_2 &= V_1 \\
y_3 &= V_2
\end{align*}
\]

\[
\begin{pmatrix} F_1 = y_2 + y_3 + e^{y_2} - 10 \\ F_2 = y_3 + y_4 - 1
\end{pmatrix} \Rightarrow \text{March}
\]

(iii) In general: given \( B_{ij}(x_1, y_1, \ldots, y_N) = 0, j=1, \ldots, \eta_1 \)

\( B_{2k}(x_2, y_1, \ldots, y_N) = 0, k=1, \ldots, \eta_2 \)

Guess \( n \) values at \( x_1 \) so as to get IVP:

\[
y_i(x_1) = f_i(x_1, V_1, \ldots, V_n), i=1, \ldots, N
\]

Discrepancy vector:

\[
F_K = B_{2k}(x_2, y_1, \ldots, y_N), K=1, \ldots, n_2
\]
(d) Shooting to intermediate point.

- Initial guess so bad that marching blows up before reaching the other end of domain;
- Integral singularities at ends that allow one to march out of, but not integrate into (see Numerical Recipes, p. 751 for more discussion).

\[ \Rightarrow \text{March from both ends to meet at some intermediate point.} \]

(i) Expand both ends into "full" BC's:
\[ y_j(x_1) = y_j(x_1, u_1, \ldots, u_{n_j}), j = 1, \ldots, N \]
\[ y_k(x_2) = y_k(x_2, v_1, \ldots, v_{n_k}), k = 1, \ldots, N \]

(Total guessed values: \( u_1, \ldots, u_{n_j}, v_1, \ldots, v_{n_k} \))

Numbering \( N \).

(ii) March both IVP's toward middle \( x_m \);

(iii) Construct \( N \)-dimensional discrepancy vector at \( x_m \):
\[ F_i = y_i(x_m, u_1, \ldots, u_{n_j}) - y_i(x_m, v_1, \ldots, v_{n_k}). \]

(iv) Root-finding for \( N \) "nonlinear equations":
\[ F_i (u_1, \ldots, u_{n_j}, v_1, \ldots, v_{n_k}) = 0 \]
by Newton's method. \( J: \text{NXN, larger system than if shooting from one end.} \)
Section 6: Finite difference solutions for B.V.P.s of ODE's

(a) General
\[ \frac{\partial y_i}{\partial x} = g_i(\bar{x}, \bar{y}_1, \bar{y}_2, \ldots, \bar{y}_N) \]
\[ B_i(\bar{x}_1, \bar{y}_1(\bar{x})), \ldots, \bar{y}_N(\bar{x}) = 0, \quad i = 1, \ldots, n_1 \]
\[ C_i(\bar{x}_1, \bar{y}_1(\bar{x}), \ldots, \bar{y}_N(\bar{x}_1)) = 0, \quad i = 1, \ldots, n_2 \]
\[ (n_1 = N - n_2) \]

For each eqn:
\[ \frac{\partial y_j}{\partial x} = g_j(x, y) \]
\[ \frac{y_j - y_{j-1}}{x_j - x_{j-1}} = g_j \left[ \frac{x_j + x_{j-1}}{2} \right] \]

Here are different ways of differencing.

To find \( N \) unknowns on each of \( M \) points: a total of \( MXN \) unknowns.

Finite differencing \( \Rightarrow \) \((M-1)XN\) equations

Boundary Conditions \( \Rightarrow \) \( N \) equations

\( \Rightarrow \) \( MXN \) coupled nonlinear algebraic eqns.

(b) A Newton-Raphson implementation:
\[ 0 = E_k = \frac{\partial}{\partial x}(x_k - x_{k-1})g_k(x_k, x_{k-1}, y_k, y_{k-1}) \]

(Vector has \( N \) elements; \( k = 2, 3, \ldots, M \))

At first boundary: \( 0 = E_1 = B(x_1, y_1) \)

Second: \( 0 = E_{M+1} = C(x_M, y_M) \)

\( E_j, \neq 0 \) for \( j = M+1, M+2, \ldots, N \)

\( E_j, \neq 0 \) for \( j = 1, 2, \ldots, M \)

This arrangement helps build a compact Jacobian matrix.
MxN equations for MxN unknowns, each equation involves only \( y_k \) and \( y_{k-1} \) (or only \( y_k \) for boundary points). This is a profound characteristic of FD equations to be exploited in solution/storage because of the "banded structure".

Newton's formula:

\[
\bar{E}_k (\bar{y}_k + \tilde{\Delta} \bar{y}_k, \bar{y}_{k-1} + \Delta \bar{y}_{k-1}) \approx \bar{E}_k (\bar{y}_k, \bar{y}_{k-1})
\]

\[
\sum_{n=1}^{N} \frac{\partial \bar{E}_k}{\partial y_{n,k-1}} \Delta y_{n,k-1} + \sum_{n=1}^{N} \frac{\partial \bar{E}_k}{\partial y_{n,k}} \Delta y_{n,k} = 0
\]

\[
\Rightarrow \sum_{n=1}^{N} \left[ \sum_{j=1}^{N} S_{j,n}^{(k)} \Delta y_{n,j-1} + \sum_{j=1}^{N} S_{j,n}^{(N)} \Delta y_{n,j} \right] = -E_{j,k}
\]

\[
S_{j,n}^{(k)} = \frac{\partial E_{j,k}}{\partial y_{n,k-1}}, \quad S_{j,n}^{(N)} = \frac{\partial E_{j,k}}{\partial y_{n,k}}
\]

Similarly:

\[
\sum_{n=1}^{N} S_{j,n}^{(1)} \Delta y_{n,j} = -E_{j,1}, \quad j = n+1, \ldots, N
\]

with \( S_{j,n}^{(1)} = \frac{\partial E_{j,1}}{\partial y_{n,j}}, \quad n = 1, 2, \ldots, N \)

\[
\sum_{n=1}^{N} S_{j,n}^{(M+1)} \Delta y_{n,m} = -E_{j,m}, \quad j = 1, 2, \ldots, n
\]

with \( S_{j,n}^{(M+1)} = \frac{\partial E_{j,m+1}}{\partial y_{n,m}}, \quad n = 1, 2, \ldots, N \)

\[
\begin{bmatrix}
S_{j,n}^{(k)} \\
S_{j,n}^{(k)}
\end{bmatrix}
\begin{bmatrix}
\Delta y_{n,j-1} \\
\Delta y_{n,j}
\end{bmatrix} = -\begin{bmatrix}
E_{j,k}
\end{bmatrix}
\]

At each point \( k \), there are \( MN \) equations, forming a "block diagonal" structure (stack the above equations for \( k = 1, \ldots, M+1 \)).
Example: 5 variables on 4 mesh points, with 3 BC’s at 1st point and 2 BC’s at 4th point.

\[
\begin{align*}
\frac{\partial E_{2,1}}{\partial y_{1,1}} & \quad \frac{\partial E_{3,1}}{\partial y_{2,1}} & \quad \frac{\partial E_{3,1}}{\partial y_{3,1}} & \quad \frac{\partial E_{3,1}}{\partial y_{4,1}} & \quad \frac{\partial E_{3,1}}{\partial y_{5,1}} \\
\frac{\partial E_{1,1}}{\partial y_{1,1}} & \quad x & \quad x & \quad x & \quad x \\
\frac{\partial E_{5,1}}{\partial y_{1,1}} & \quad x & \quad x & \quad x & \quad x \\
\frac{\partial E_{1,2}}{\partial y_{1,1}} & \quad \frac{\partial E_{1,2}}{\partial y_{2,1}} & \quad \frac{\partial E_{1,2}}{\partial y_{3,1}} & \quad \frac{\partial E_{1,2}}{\partial y_{4,1}} & \quad \frac{\partial E_{1,2}}{\partial y_{5,1}} & \quad \frac{\partial E_{1,2}}{\partial y_{6,1}} \\
\end{align*}
\]

Show block diagonal structure on the following pages.

(c) A Gauss elimination scheme: operate by blocks.

See Fortran code in §17.3 of N.O.R.

Details of coding: * Pivoting — ordering of eqns.
* Storage
* Convergence criterion
Figure 17.3.1. Matrix structure of a set of linear finite-difference equations (FDEs) with boundary conditions imposed at both endpoints. Here $X$ represents a coefficient of the FDEs, $V$ represents a component of the unknown solution vector, and $B$ is a component of the known right-hand side. Empty spaces represent zeros. The matrix equation is to be solved by a special form of Gaussian elimination.

(See text for details.)

Figure 17.3.2. Target structure of the Gaussian elimination. Once the matrix of Figure 17.3.1 has been reduced to this form, the solution follows quickly by backsubstitution.
Figure 17.3.3. Reduction process for the first (upper left) block of the matrix in Figure 17.3.1. (a) Original form of the block, (b) final form. (See text for explanation.)

(a) \[
\begin{array}{cccc}
D & D & D & A \\
D & D & D & A \\
D & D & D & A \\
\end{array}
\]

(b) \[
\begin{array}{cccc}
0 & 0 & 0 & S \\
0 & 1 & 0 & S \\
0 & 0 & 1 & S \\
\end{array}
\]

Figure 17.3.4. Reduction process for intermediate blocks of the matrix in Figure 17.3.1. (a) Original form, (b) final form. (See text for explanation.)

(a) \[
\begin{array}{cccccccc}
0 & 0 & 0 & 1 & 0 & 0 & 0 & S \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & S \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & S \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & S \\
Z & Z & Z & D & D & D & D & A \\
Z & Z & Z & D & D & D & D & A \\
Z & Z & Z & D & D & D & D & A \\
Z & Z & Z & D & D & D & D & A \\
\end{array}
\]

(b) \[
\begin{array}{cccccccc}
0 & 0 & 0 & 1 & 0 & 0 & 0 & S \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & S \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & S \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & S \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & S \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & S \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & S \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & S \\
\end{array}
\]

Figure 17.3.5. Reduction process for the last (lower right) block of the matrix in Figure 17.3.1. (a) Original form, (b) final form. (See text for explanation.)