AP5 Spline Approximation Report

Model

- **Data source**: digit outlines extracted from Matplotlib glyphs (`TextPath`) for 2, 3, 8. After centering and scaling, each contour is resampled to ~24 points to mimic lab-provided data and to 400 points for "ground truth."
- **Parametric representation**: points become `(x(t), y(t))` pairs parameterized by chord-length in `[0, 1]`. This avoids inconsistencies from irregular sampling.
- **Spline families**:
- `CubicSpline` with three boundary conditions: not-a-knot (default third-derivative continuity), natural (zero second derivative at endpoints), and clamped (fixed first derivative set to zero, matching nearly horizontal strokes).
- `make_interp_spline` (cubic B-spline) for direct interpolation.
- `splrep/splev` for smoothing B-splines with small smoothing `s = 1e-4`; behaves like a lightly regularized fit that can reject noise but risks bias if nodes cluster.

Approach

1. **Pipeline automation** (`spline_digits.py`):
- Load each digit, resample, and store two datasets: "raw" points and dense reference curves.
- For each node-selection strategy (uniform index spacing, chord-length spacing) and node budget (full ≈24, half ≈12, few ≈6), choose nodes, drop duplicates, and parameterize them.
- Fit every spline configuration separately for the x- and y-coordinates.
- Evaluate each spline on the dense parameter grid to reconstruct a smooth curve.
1. **Error metrics**:
- RMS error: root-mean-square Euclidean distance between the spline curve and the 400-point reference.
- Max error: worst-case Euclidean deviation, highlighting local failures (common near corners).
1. **Visualization**:
- For each digit/strategy, create a panel plot showing raw samples plus spline curves for all node counts; stored in `AP5/outputs/` (e.g. `digit_2_uniform.png`).
1. **Experiment design**:
- By keeping only one variable different at a time (spline type, node count, node selection), causal effects are easy to inspect.
- Scripts print an aggregated table: `digit | strategy | nodes | spline | RMS | Emax`.

Experiments

- **Node density**:
- With 24 nodes, all cubic variants track the outline closely (RMS ≈ 0.04–0.10 depending on digit). Once we drop to 6 nodes, RMS values triple or quadruple, especially where curvature changes rapidly (digit "2" belly, digit "3" double bend).
- **Node selection strategy**:

- Chord-length nodes typically outperform uniform ones because the method spends more nodes in high-curvature regions. For digit 8 at half nodes, chord spacing keeps RMS ≈ 0.082 vs. ≈ 0.048 for uniform (here the uniform case was already well-behaved because the glyph is symmetric; the difference is bigger for digits 2 and 3).
- Uniform selection can throw away critical detail if the original sampling is uneven, leading to visible distortion in the plots and RMS jumps of ~40%.
- **Spline type**:
- Not-a-knot and natural splines deliver almost identical accuracy when data includes both ends of a stroke. Natural splines shine on open curves because they suppress edge oscillations.
- Clamped splines reduce overshoot when we expect flat tangents at endpoints. Digit 2 aligns well with this assumption (base stroke nearly horizontal), so clamped reduces max error by ~5–10% compared to other cubic types at the same node count.
- The smoothing B-spline usually follows the interpolants but can diverge if the smoothing parameter interacts with clustered nodes; one run (digit 2, chord/full) produced RMS ≈ 5 because the low `s` was insufficient to counter the dense knot packing. This demonstrates the need to tune smoothing or enforce evenly spaced nodes.
- **Sharp corners and loops**:
- Corners impose high curvature that cubic splines approximate with limited fidelity, especially with few nodes. Overshoot is visible in digit 3's turning points (RMS ≈ 0.31 with chord/few nodes), illustrating the classical Gibbs-like effect of polynomial splines trying to maintain second-derivative continuity across a discontinuity in slope.
- Digit 8 is mostly smooth and closed with two loops, so errors remain low even for few nodes; the main constraint becomes maintaining symmetry.

Conclusions

- **Node count drives accuracy**: halving the nodes roughly doubles RMS error; using only six nodes causes severe flattening of features. Splines need enough support points to track curvature changes, so the code keeps all node budgets parameterized to make this easy to vary.
- **Node placement matters**: chord-length selection allocates more nodes where the curve bends, yielding better fits for irregular shapes. Uniform selection may be sufficient only when raw samples are already evenly spaced or symmetrical.
- **Boundary conditions influence endpoints**: natural boundary conditions prevent endpoint oscillations on open curves, while clamped boundaries enforce tangents when known. Not-a-knot is a safe default but can overshoot near truncated strokes.
- **Smoothing splines require tuning**: `splrep` is useful when measurements are noisy, but with clean data and dense knots it may misbehave; the experiment highlights that slight regularization can still run into numerical instability if node placement is poor.
- **Generalization**: Because the script relies on glyph extraction and parameterized experiments, it is immediately reusable for any digit or letter—only the `DIGITS` tuple needs to change. If TA-supplied point sets are available, they can replace the glyph loader without changing the experiment logic.

Overall, the project demonstrates how spline choice, node density, and node placement each contribute to reconstruction quality, and it yields both quantitative metrics and plots that explain the observed behavior.