Academic Paper: Development of a Tangential Control System for CNC Machines Using State Tags in LinuxCNC

Abstract

Continuous tangential motion of cutting tools (e.g., tangential knives for leather/fabric cutting, welding heads, and textile tools) represents a fundamental challenge in open-source CNC control systems. The LinuxCNC environment suffers from a structural separation between the Interpreter (running in user space) and the Motion Controller (running in real-time). This separation leads to inaccuracies in tangential angle calculation at transition points or during axis stops. This paper presents a solution based on **State Tags** to transfer geometric information (straight-line heading, arc center, and radius) from the Interpreter to the Motion Controller. For straight lines (G1), the tangential angle is computed once in the Interpreter using the `atan2` function. For circular arcs (G2/G3), the angle is computed in real-time within the Motion Controller using the current tool position and the arc center. The required code modifications are analyzed in detail for the core files: `interp_convert.cc`, `interp_write.cc`, `state_tag.h`, and `control.c`. Critical vulnerabilities such as queue data loss and mathematical singularity are identified. The paper concludes with practical recommendations to ensure stability and integration into the main LinuxCNC branch.

Keywords: Tangential control, LinuxCNC, State Tags, real-time, circular arcs, command queue.

1. Introduction

Computer Numerical Control (CNC) systems are the backbone of modern manufacturing. Among specialized applications, there is a growing need for tools that maintain a precise tangential orientation relative to the toolpath. Examples include oscillating knives for cutting leather and fabrics, welding heads for pipe production lines, and textile layup tools. In such applications, the rotary axis (often the A or C axis) must remain perpendicular to the path at every instant.

LinuxCNC is a flexible, open-source CNC platform. However, its internal architecture separates two critical components:

- **The Interpreter (user space):** Reads G-code, performs geometric calculations (e.g., arc centers, radii), and places commands into a queue for execution.

- **The Motion Controller (real-time kernel space):** Executes commands as successive waypoints at high frequency (typically 1 kHz), but it has no knowledge of the full path geometry – it is geometrically "blind".

This separation causes traditional tangential calculations based only on instantaneous differences $(\Delta x, \Delta y)$ to fail, especially when the machine stops (differences become zero and the angle is lost) or at sharp corners. This paper presents an integrated mathematical and engineering solution, along with a detailed code analysis of required modifications in the LinuxCNC source tree, building upon technical discussions with developer Rod Webster and PR #900.

2. Problem Statement

Consider a compound motion consisting of a straight segment followed by a circular arc and then another straight segment. The rotary axis (A-axis) must maintain a precise tangential heading at every instant.

Core problem: In its current state, LinuxCNC cannot compute the correct tangential angle for the following reasons:

1. **The Interpreter** knows the geometry but does not execute in real-time.
2. **The Motion Controller** executes in real-time but does not know the geometry.
3. Any attempt to compute the angle from consecutive waypoints inside the Motion Controller suffers from:
   - **Lag:** The computed angle corresponds to the previous segment, not the current instant.
   - **Zero-speed failure:** When the X/Y axes stop, $\Delta x = \Delta y = 0$ and the angle becomes undefined.

Therefore, geometric information must be transferred from the Interpreter (where it is available) to the Motion Controller (where it is needed) using a reliable mechanism – **State Tags**.

3. Mathematical Approach

We distinguish two motion types: straight lines and circular arcs.

3.1 Straight Lines (G1)

The tangential angle is constant along the entire straight segment. It is computed once inside the Interpreter using the start and end point coordinates:

$$\theta_{\text{straight}} = \text{atan2}(y_{\text{end}} - y_{\text{start}},\ x_{\text{end}} - x_{\text{start}})$$

where `atan2` is the inverse trigonometric function returning an angle in the range $(-\pi, \pi]$. This value is stored in the `GM_FIELD_FLOAT_STRAIGHT_HEADING` State Tag and sent to the Motion Controller as a constant value for the entire G1 command.

3.2 Circular Arcs (G2/G3)

For arcs, the tangential angle changes continuously. It must be computed in real-time inside the Motion Controller using geometric information sent from the Interpreter (arc center and radius). The required equations are as follows.

Inputs from the Interpreter (via State Tags):

- Arc center: $C(X_c, Y_c)$
- Radius: R (optional, can be derived from center and start point)
- Direction: clockwise (G2) or counter-clockwise (G3)

Inputs available in the Motion Controller at each servo cycle:

- Current tool position: $P(X_p, Y_p)$

Radius vector (from center to current point):

$$\vec{N} = [X_p - X_c,\ Y_p - Y_c]$$

Angle of this radius vector (relative to X-axis):

$$\alpha = \text{atan2}(Y_p - Y_c,\ X_p - X_c)$$

Tangential angle is perpendicular to the radius vector, therefore:

$$\theta_{\text{arc}} = \alpha \pm \frac{\pi}{2}$$

The sign is chosen as follows:

- $+\frac{\pi}{2}$ for counter-clockwise arcs (G3)
- $-\frac{\pi}{2}$ for clockwise arcs (G2)

The exact sign convention depends on the definition of the rotary axis direction, but the principle remains consistent.

3.3 Singularity Handling

If the current tool position P coincides with the arc center C (a rare but possible situation, e.g., at the very start of a full circle or due to a programming error), the radius vector becomes zero and `atan2(0,0)` is undefined. To protect the system, we add a conditional safeguard:

$$\text{if } (X_p - X_c = 0 \text{ and } Y_p - Y_c = 0) \quad \Rightarrow \quad \theta_{\text{arc}} = \theta_{\text{previous}}$$

That is, retain the last valid tangential angle. In practice, a small tolerance (e.g., 1×10^{-12}) is used instead of exact equality.

4. Programming Analysis

4.1 Files and Functions Involved

Based on the LinuxCNC architecture [1], modifications span three layers.

a) Interpreter Layer

- **File:** `src/emc/rs274ngc/interp_convert.cc`
  - **Function** `convert_straight(...)` **:** Called for G1 commands. After computing the heading, it must invoke the State Tag mechanism to store the value.
  - **Function** `convert_arc(...)` **or** `convert_arc2(...)` **:** Called for G2/G3 commands. Must store the arc center `(I, J, K)` and radius `R` into State Tags.
- **File:** `src/emc/rs274ngc/interp_write.cc`
  - **Function** `write_canon_state_tag(block, &_setup)` **:** This is the critical function that writes State Tags to shared memory. The reported problem is that this function is sometimes not called for every motion command, leading to data loss in the queue.

b) State Tag Definition Layer

- **File:** `src/emc/rs274ngc/state_tag.h`

New fields must be added:

```
c
```

```
// New fields for tangential control
GM_FIELD_FLOAT_ARC_RADIUS,
GM_FIELD_FLOAT_ARC_CENTER_X,
GM_FIELD_FLOAT_ARC_CENTER_Y,
GM_FIELD_FLOAT_STRAIGHT_HEADING
```

c) Motion Controller Layer (Real-time)

- **File:** `src/emc/motion/control.c`
  - **Proposed function (or integrated into the main control loop):** Reads the State Tag associated with the current motion command.
  - If the command is G1: directly uses `STRAIGHT_HEADING`.
  - If the command is G2/G3: executes the real-time angle calculation (Section 3.2) every servo cycle.
  - Outputs the final angle to a HAL pin (e.g., `motion.rotary-angle-cmd`) that drives the tangential axis.

4.2 Code Gaps and Risks

Risk	Description	Severity	Recommendation
Queue data loss	If `write_canon_state_tag` is not called for every motion command, data "slips": the Motion Controller applies the heading of a previous G1 to a following G2 arc, causing erratic behavior.	**High**	Create a mandatory wra function that guarantee `write_canon_state_tag` called at the end of eve motion-generating func
Mathematical singularity	When `X_p = X_c` and `Y_p = Y_c`, `atan2(0,0)` is undefined.	**Medium**	Implement the safeguar shown in Section 3.3 wi small epsilon tolerance.
Real-time performance	Computing `atan2` every servo cycle (e.g., every 1 ms) may add a small CPU overhead.	**Low**	Use a fast, low-precisio version like `atan2_fast` available in the kernel environment, or rely on standard library which i usually optimized.

4.3 Proposed Code Snippet for `control.c`

```c
// Inside the main control loop (simplified)
static double previous_tangential_angle = 0.0;

double compute_tangential_angle(EmcPose *current_pos, CanonStateTag *tag) {
    double theta;
    const double epsilon = 1e-12;

    if (tag->motion_type == G1_STRAIGHT) {
        theta = tag->straight_heading;
    }
    else if (tag->motion_type == G2_ARC || tag->motion_type == G3_ARC) {
        double dx = current_pos->x - tag->arc_center_x;
        double dy = current_pos->y - tag->arc_center_y;
        if (fabs(dx) < epsilon && fabs(dy) < epsilon) {
            // Singularity: keep previous angle
            theta = previous_tangential_angle;
        } else {
            double alpha = atan2(dy, dx);
            double sign = (tag->motion_type == G3_ARC) ? 1.0 : -1.0;
            theta = alpha + sign * M_PI_2;
        }
    }
    else {
        // Default: no change
        theta = previous_tangential_angle;
    }

    previous_tangential_angle = theta;
    return theta;
}
```

4.4 Integration with HAL

The computed angle must be written to a HAL pin. The conventional pin for rotary axis command is:

```c
hal_float_t *rotary_angle_cmd;  // e.g., motion.rotary-angle-cmd
// In control loop:
*rotary_angle_cmd = theta;
```

Alternatively, a dedicated pin like `motion.tangential-angle-cmd` can be created.

5. Results and Recommendations

5.1 Expected Results

Upon implementing the proposed solution, LinuxCNC will be able to:

- Maintain a precise tangential angle even when the linear axes are stopped (because the angle is stored from the Interpreter).
- Handle circular arcs smoothly without jerks or discontinuities.
- Avoid synchronization errors between Interpreter and Motion Controller thanks to the reliable State Tag mechanism.

5.2 Recommendations for Contributing to PR #900

To help successfully complete Pull Request #900 (submitted by Rod Webster), the following steps are recommended:

1. **Build an integration test:** Create a simple G-code file with the sequence `G1 → G2 → G1`. Use `halscope` to monitor the tangential axis output signal and verify that the angle transitions smoothly at the connection points.
2. **Audit `write_canon_state_tag` calls:** Trace all invocations in `interp_write.cc` to ensure that every `convert_straight` and `convert_arc` call leads to a corresponding State Tag write.
3. **Document mathematical equations** inside the source code using LaTeX-style comments to aid future maintainers.
4. **Test the singularity condition:** Simulate a scenario where the arc start point coincides with the arc center (unusual but possible) to verify that the safeguard works without crashing or producing `NaN`.

5.3 Conclusion

The tangential control problem in LinuxCNC is not a mathematical deficiency but rather a communication design issue between software components. By using **State Tags** as a bridge to transfer geometric information from the Interpreter (where it is easily available) to the Motion Controller (where it is needed in real-time), high accuracy can be achieved without sacrificing real-time performance. The programming analysis presented here precisely identifies the files and functions requiring modification and warns about the main risks (queue data loss and mathematical singularity). We recommend that the

LinuxCNC community merge these modifications into the master branch, as they will enable advanced manufacturing applications that were previously difficult or impossible.

References

[1] LinuxCNC Documentation Team. (2024). *LinuxCNC Developer Notes*. Retrieved from https://linuxcnc.org/docs/devel/html/code/code-notes.html

[2] Webster, R. (2023). *Tangential Control using State Tags (PR #900)*. LinuxCNC GitHub Repository.

[3] Proctor, F., & Michaloski, J. (2015). *Real-time Motion Control for Open Architecture CNC*. NIST Interagency Report 8082.