

From Benchmarked Beam Solver to Practical S-Lay Overbend Tool: An LLM-Assisted FEA Programming Case Study

Sreekanth Manakkattil Sivaraman

Principal Subsea Structural Engineer

<https://www.linkedin.com/in/sreekanth-sivaraman-71b5a49/>
sreekanth.sivaraman@gmail.com

Abstract

This paper presents an application-level case study in which an existing benchmarked nonlinear beam finite-element solver is adapted into a practical S-lay overbend analysis program for offshore pipelines and subsea inline components, using a large language model (LLM) as the main programming and software-structuring assistant. The mathematical development of the nonlinear beam solver and of its plasticity extension was completed in earlier work and is not revisited. The present study addresses the next engineering problem: how a validated solver kernel is converted into a usable domain-specific analysis tool by implementing geometry preparation, roller contact, lay tension, staged loading, component passage sweeps, result extraction, validation ledgers, plotting and machine-learning dataset generation.

The central argument is that once a finite-element formulation has been established and benchmarked, many subsequent engineering software tasks become physics-implementation problems rather than new mathematical derivations. In such tasks a domain expert can specify the physical meaning of the model, the acceptance criteria and the benchmark comparisons, while an LLM performs most of the implementation, refactoring, documentation and regression-test generation. The case study is S-lay overbend analysis of plain pipe, thickened pipe components, offset shrouds and externally attached pipeline-mounted structures.

The rebuilt program is compared against two published S-lay inline-structure studies over **57 published comparisons drawn from 50 published cases**, every one produced by a single build of the program on a single date. The study shows that LLMs can substantially reduce the programming barrier for domain experts who need to customise validated numerical solvers into practical engineering tools. It also shows what that costs: of the five defects the benchmark campaign exposed, none was in the solver and none would have been found by a test the LLM could have written unprompted. The LLM can implement the engineering workflow; the domain expert must define and validate the physics.

Keywords: nonlinear beam finite element, S-lay overbend analysis, large language model, engineering software development, machine-learning dataset generation

Short name	Work
Stage I	The earlier nonlinear beam solver development, Ref [4]
Stage II	The earlier plasticity extension, Ref [5]
Reference Study 1	Ref [1]
Reference Study 2	Ref [2]
EDES / EDAS	The component layer and the assembly layer of , Ref [3]

Published tables are cited by the Roman numerals those studies give them — TABLE X, TABLE XXXI and so on — which are unambiguous once the study is named. Reference Study 1 supplies the plain-pipe, thick-pipe and shroud cases; Reference Study 2 supplies the externally attached cases.

The two reference studies, and the knowledge-graph work, are by the present author. This comparison is therefore a reproduction of the author's own earlier commercial-FEA results by an independently rebuilt program, not an independent third-party validation, and Section 11 treats that as a limitation rather than a detail.

1. Introduction

Finite-element solvers are often developed and benchmarked as mathematical tools, but practical engineering use requires a further layer of work. Engineers rarely use a bare solver directly. They need a problem-specific program that converts engineering inputs into an analysis model, applies realistic boundary conditions and loading sequences, extracts results in the form a design decision needs, and checks those results against known benchmarks.

This paper studies that second layer. The finite-element kernel already exists: a nonlinear beam solver previously developed and benchmarked using an LLM-assisted workflow [4], then extended to material plasticity in a second benchmarked study [5]. The current work asks a different question:

Can a domain expert, with the help of an LLM, adapt an existing benchmarked nonlinear beam solver into a practical engineering analysis tool without re-deriving the solver mathematics — and what has to be true of the workflow for the result to be believable?

The engineering application is S-lay overbend analysis of offshore pipelines and inline components. The tool must represent vessel and stinger rollers, pipe self-weight, lay tension, plastic bending, one-sided contact, inline component stiffness, shroud lift and the passage of a component across the overbend region. These are not primarily new finite-element derivations. They are modelling, workflow and validation problems.

2. Position of this work relative to the earlier solver stages

This paper does not claim that an LLM can safely invent finite-element mathematics without validation. The strict solver work belongs to the earlier stages.

Stage I developed a Python nonlinear beam finite-element solver for large-displacement analysis, focusing on formulation, element behaviour, nonlinear solution and benchmark comparison against recognised beam problems. Stage II extended that solver to material plasticity, including Ramberg–Osgood and incremental J2 plasticity, fibre integration and comparison against commercial FEA results. Together they establish the solver kernel, which the current work uses as a reusable engine.

The kernel is frozen. It is carried in the repository as a single module and the project’s guardrails forbid editing it: a defect traced into the kernel must be reported and worked around at the application layer, not patched, because the moment it is patched it is no longer the thing the earlier stages benchmarked. No defect found in this study required it. That is itself a result — the five defects of Section 7 all lie above the kernel.

Table 1 — Relationship between the three stages of work

Stage	Main problem	Main evidence	Role in the current work
Stage I, [4]	Nonlinear beam solver development	Large-displacement beam benchmarks	Provides the base solver kernel
Stage II, [5]	Plasticity extension	Material plasticity and commercial-FEA comparison	Provides nonlinear material capability
This paper	S-lay engineering application	Pipeline and inline-component benchmark studies; 57 published comparisons	Demonstrates solver customisation into a practical tool

Stage 1	Stage 2	This Paper
•Nonlinear beam solver •Co-rotational beam elements, large rotation •Benchmarked against NAFEMS	•Plasticity Extension •Multi-linear material, yielding and unloading •Validated against Industrial Software	•S-lay overbend application •Stinger geometry, roller contact, inline components, passage sweep •Validated against Industrial Software

Figure 1. Solver-to-application progression. The mathematics is inherited from the two earlier stages; the physical idealisation and the application layer are the contribution of this work.

3. Problem reframing: from mathematical solver to physics implementation

Once the solver kernel is available, the engineering work shifts from deriving stiffness matrices to representing the practical problem correctly. For S-lay overbend analysis the application-layer questions are:

- What is the physical datum of the stinger radius?
- Which nodes may contact a roller?
- Does roller contact follow the structural line, or the component ownership label?
- Is lay tension attached to a material point, or applied at a station the pipe slides through?
- Has the component completed its required passage across the stinger?
- Which strain region corresponds to the published benchmark quantity?
- Is a converged solution physically valid if it leaves the material data range?

These are not mathematically trivial, but they are not new finite-element theory either. They are physics-implementation questions, and this is the zone where LLM assistance is most useful: the expert describes the physical behaviour and the validation requirement, and the LLM translates it into code, tests, data structures, runners and documentation.

It is also, as Section 7 shows, exactly the zone where the defects were. Every one of the seven questions above was answered wrongly at some point in this project, each time by code that converged, ran to completion and printed a plausible number.

4. Methodology: domain expert plus LLM

4.1 Division of responsibility

The LLM was used as a programming collaborator, not as the final engineering authority. The domain expert controlled the physical modelling assumptions, the interpretation of the S-lay installation process, benchmark case selection, recognition of non-comparable results, acceptance or rejection of validation outcomes, and the decision on whether a discrepancy was acceptable, open or invalidating. The LLM performed or assisted with code implementation, refactoring older scripts into layered modules, command-line runner generation, plotting and result schemas, test generation, documentation, validation-table preparation and the diagnosis of possible code-level causes.

This division is central to the paper. The LLM can do much of the programming, but the engineering meaning of the program remains the responsibility of the domain expert.

Table 2 — Domain expert and LLM responsibility split

Activity	Domain expert	LLM
Solver mathematics	Defines acceptance from the earlier benchmarked work	Reuses the existing implementation unchanged
Physical modelling	Defines contact, load and component assumptions	Implements the assumptions in code
Benchmark selection	Chooses the relevant published cases	Builds runners and comparison tables
Result interpretation	Accepts, rejects or marks open	Summarises and organises the evidence
Software structure	Reviews architectural suitability	Refactors and enforces layer boundaries
Regression testing	Defines the required behaviour	Writes the tests and runs the checks

4.2 Controls used

The workflow used eight controls intended to stop LLM output becoming unverified code. Each is named here with what it is actually capable of catching, because a control whose reach is not stated tends to be credited with more than it does.

Control	What it catches	What it cannot catch
Layered architecture, nine layers	A module reaching across the design	Wrong physics inside a correctly placed module
Automated layer-import checker	Any violation of the above, on every run	—
Frozen solver kernel	A defect being “fixed” where it would invalidate Stage I and II	—
Mirrored component definitions, hashed	The component specification drifting from its source	—
Versioned validation ledger	A number whose build or date is unknown	A number that is wrong in a way no benchmark covers
Defect and lesson register, 132 entries	The same mistake twice	The first instance
Explicit completion status on every passage	A truncated traverse compared against a complete published one	—
A regression test for every defect found	That specific defect returning	A defect of a kind not yet seen

The honest summary of that table is in its right-hand column. The controls are good at preventing recurrence and at making provenance auditable. They found none of the five defects in Section 7. Benchmark comparison found all five.

5. S-lay program architecture

The application program is organised as a layered toolchain:

data → define → scene → model → physics → solve → study → report → entry

A module may import only from the layers to its left. This is enforced mechanically rather than by convention: a checker walks every module in the application package on every run and fails on any violation. At the time of writing the package holds **40 modules** inside those nine layers — the number the checker reports, which excludes the package's two un-layered files — with **46 test modules** beside them and **31 command-line tools** above them. The solver kernel sits below the whole stack and is not part of it.

Table 3 — Program layers and responsibilities

Layer	Responsibility
data	Static input data, section tables and material definitions
define	Component archetypes, and their reduction to simplified representations
scene	Stinger geometry, roller stations and the installation scene
model	Mesh, sections, component bodies and ownership
physics	Contact targets, loads, material bindings and the problem object
solve	Nonlinear solution, active-set contact and the passage sweep
study	Passage sweeps, staged runs and parameter studies
report	Envelopes, regions, station windows, artifacts and validation output
entry	Command-line tools and reproducible runners

5.1 Building the inline structure: component definitions and assembly rules

An inline structure is not described to this program as a mesh, and not as a set of nodes and elements. It is described as **what it is** — a list of components with their own dimensions, plus the rules by which those components may be assembled onto a pipeline. Turning that description into something an analysis can use is a separate layer of work with its own failure modes, and it is done before any stinger, roller, load or solver exists.

This follows the **EDES / EDAS** scheme set out in the authors' earlier knowledge-graph work on S-lay inline structures [3], where engineering design knowledge is split into a component layer and an assembly layer:

- **EDES**, the component layer, holds what each component *is*: its class (thickened pipe body, tapered transition, offset shroud, top structure, base structure, valve, branch piping, connector), the parameters that dimension it, the named features on it that other things may attach to — weldL, weldR, conMid, tee, end, pipeEnd, structEnd, slot1...slot5 — and the default for every parameter not given.
- **EDAS**, the assembly layer, holds what may be *done* with them: which anchors are permitted, which associations are required, which component may support or protect which other, and how a declared connection system resolves into occupied slots.

The builder creates a validated two-dimensional layout of the inline structure in its own local coordinate system. The origin is at the structure reference point, and +x points toward the vessel. At this stage, the layout does not know anything about the stinger, rollers, mesh, loads or solver. This separation is intentional. It allows the structure layout to be checked, plotted, stored and reused before it is placed into an installation analysis model.

The builder handles the assembly-level checks that individual components cannot do on their own. It checks that component dimensions are compatible, calculates the overall extent and mass, resolves the connection system, and decides which body controls contact where components overlap.

For validation, the builder samples the full length of the structure and asks two questions at many locations:

What structural section is present here?

What surface would a roller touch here?

The section depth and contact depth are not always the same. A thickened pipe changes the pipe section, and the roller contact surface follows the thicker outside wall. An offset shroud does not change the pipe section; it only moves the contact surface outward. A base structure also does not change the pipe section, but it changes the load path: roller load reaches the pipe through discrete connectors rather than directly through the pipe wall. This is why the base-structure cases behave differently from thickened pipe and shroud cases.

If the layout violates the assembly rules, the builder rejects it and reports the rule that failed. For example, the base-structure archetype used in this study has a two-slot connection system but does not place separate connector parts at those slots. The builder reports this warning explicitly instead of silently accepting an unclear layout.

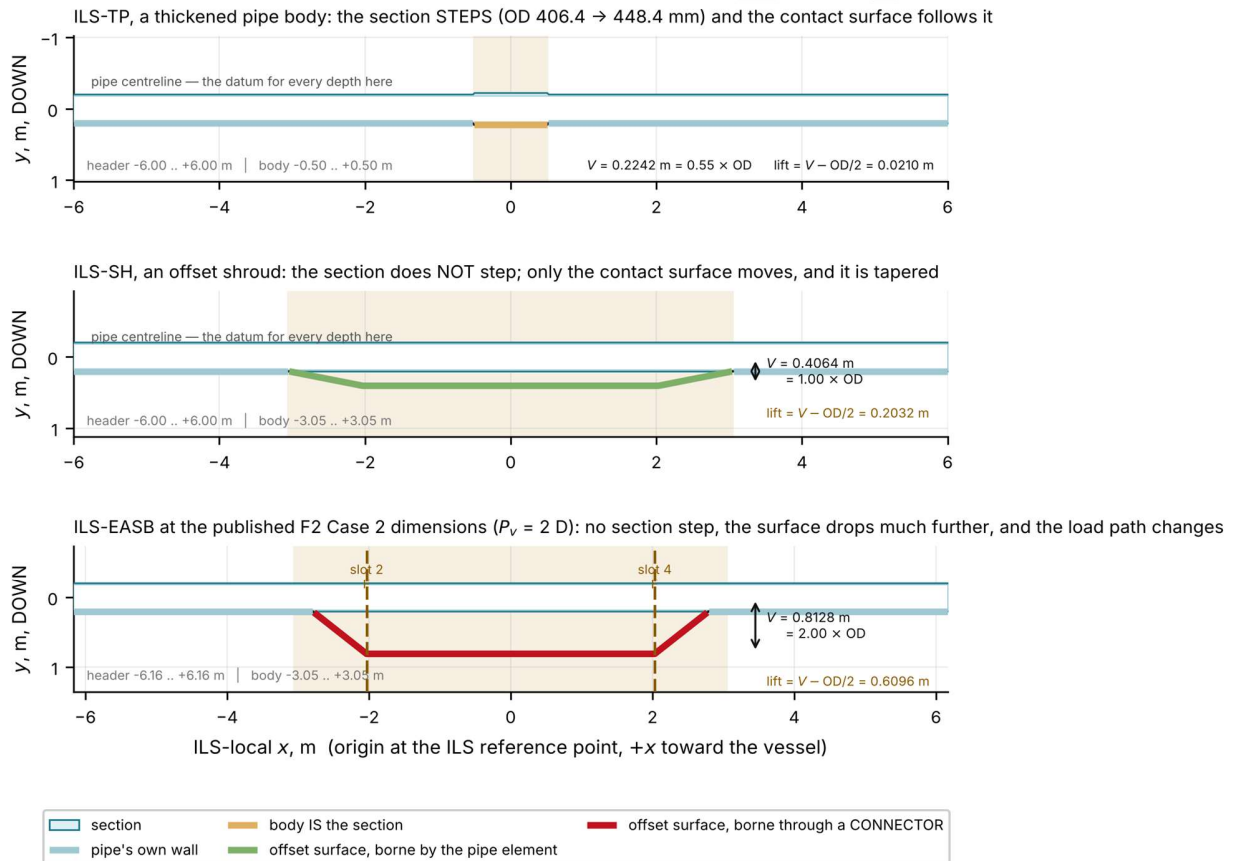


Figure 2. The same two questions, asked of three archetypes and answered differently by each. The pale band is the section the assembly reports; the heavy line beneath it is the contact surface, coloured by load path; the dashed horizontal line is the pipe centreline, which is the datum for both. The base-structure panel is dimensioned as the published F2 Case 2, with its depth V and the lift $V - \text{OD}/2$ marked as measured quantities. Drawn from the builder's own answers sampled at 601 positions across each header, not from the definitions that were fed to it.

5.2 Placing the structure in the lay model

The layout is placed into the S-lay model only after it passes validation. Placement is just a coordinate shift. A local structure coordinate x_{local} becomes a model station:

$$s = s_{\text{centre}} - x_{\text{local}}$$

The minus sign is needed because the local structure +x direction points toward the vessel, while the model +s direction points toward the stinger tip.

Keeping the structure in its own local frame until this step makes the design movable. The same validated layout can be placed at different positions along the pipeline without rebuilding the component definition.

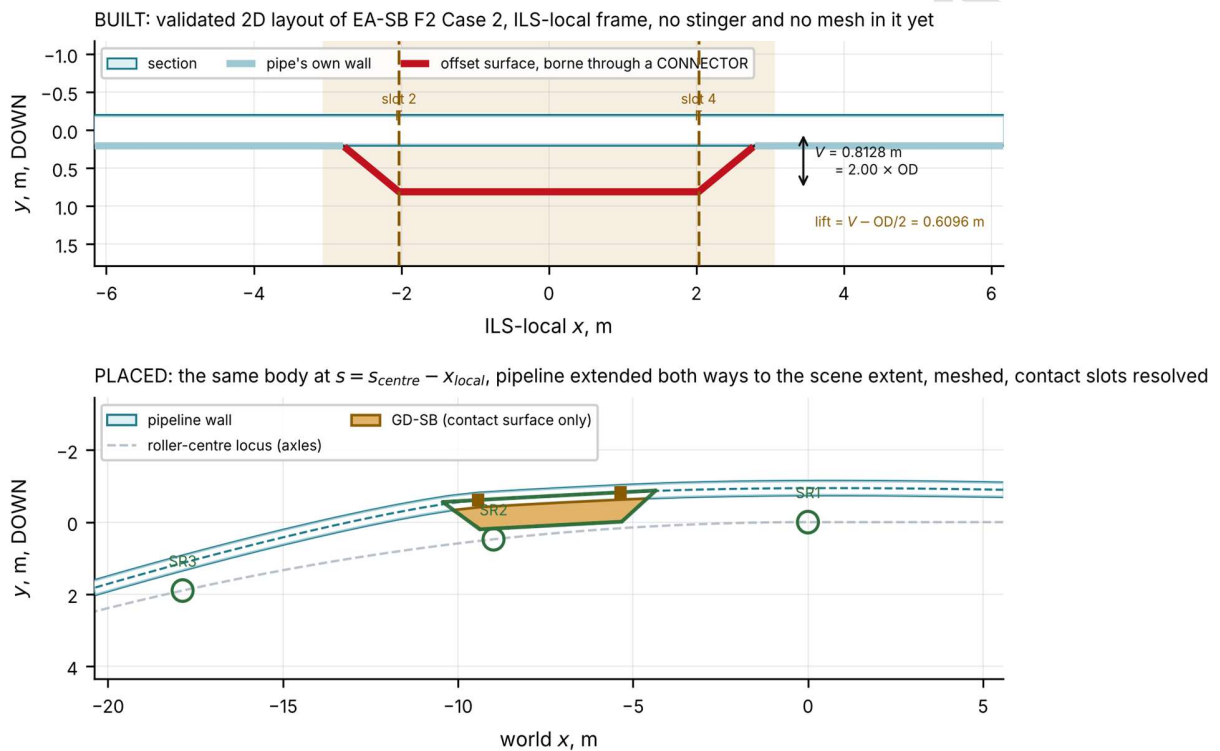


Figure 3. Built, then placed — EA-SB F2 Case 2 in both panels, at the same dimensions. Above: the validated layout in its own frame, with nothing of the lay system in it, and the depth $V = 2 D$ measured from the pipe centreline. Below: the same body on the stinger at true scale, pipeline extended both ways, meshed, resting on a roller through the contact surface the assembly resolved. The two brown squares are the connectors through which this family's load reaches the pipe.

After placement, the model is completed as follows:

1. Extend the pipeline on both sides of the structure, so the analysis represents an inline structure installed in a pipeline, not the structure alone.
2. Force nodes at important stations, such as component ends, connector pipe-side locations, elastic-zone boundaries, load points and restraint points. This is needed because point loads, restraints and connector forces must act at nodes.
3. Mesh the pipeline and structure spans according to the sections defined by the layout.

4. Resolve the contact surface along the full line. The roller contacts the structure surface where the structure controls contact, and the pipe outside wall elsewhere. If two bodies overlap, only the lowest contact surface is used to avoid double counting.
5. Create roller contact targets at the roller stations, using the resolved contact surface rather than the pipe centerline.

The key point is that contact is an assembly property, not just a component label. This mattered in the validation. Earlier, contact was selected using the component reporting label, which caused long component body nodes to be excluded from roller contact. As a result, the model could not reproduce the published multi-roller engagement.

5.3 Stinger and contact geometry

The stinger radius is measured to the **roller centreline**. The roller top is one roller radius above that, and the pipe's bottom surface rests on the roller top, so the pipe centreline rides at

$$R_{\text{eff}} = R + r_{\text{roller}} + D/2$$

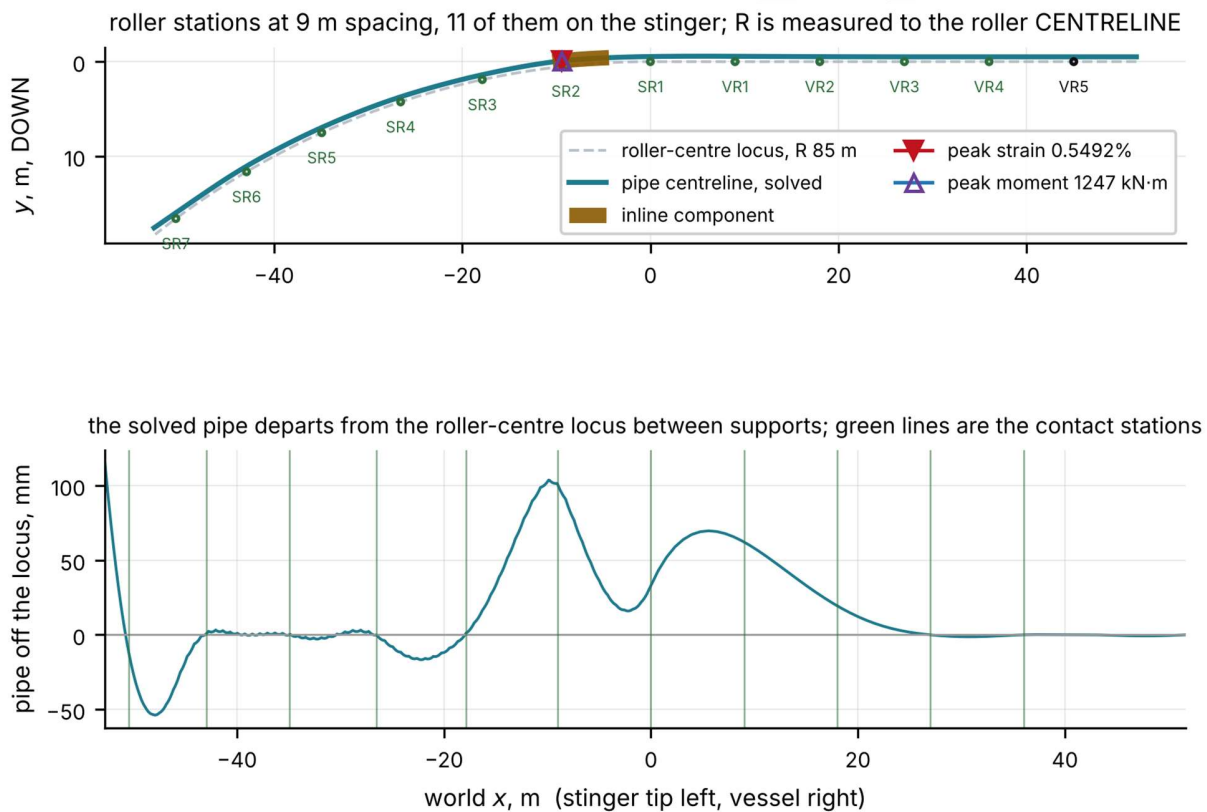


Figure 4. Above: the roller stations and the arc R is measured to, with the solved pipe above it. Below: how far the pipe departs from that arc along the same case. The pipe does not ride on the locus the radius is measured to, and the contact stations do not fall on nodes — both are deliberate, and both are things a figure drawn from a formula rather than from the solved model would get wrong.

This avoids forcing the pipe centerline directly onto the roller-centerline arc. The correction is applied through the effective radius, rather than by adding a vertical or normal offset to every contact target. This is important because a uniform offset along the straight deck should not create bending strain. By applying the correction through the radius, only the curved stinger region is affected, which matches the intended physical behavior.

5.4 Passage sweep

The component passage study shifts the component relative to the stinger and records the envelope of the response. This requires the program to distinguish **material coordinates** — a point on the pipe, which moves — from **station coordinates** — a point on the vessel, which does not. Lay tension, roller contact and component location must each move consistently with respect to the right one of those two.

The reported value for a case is therefore not a single analysis. It is a maximum over a traverse of independently solved nonlinear positions, and this has a consequence for every comparison in Section 6: an envelope cannot read lower than a single position of the same configuration. Where the current tool reads *high* against a published single-position value, part of the difference may be the envelope. Where it reads *low* — as the externally attached base structures do — the envelope is eliminated as the explanation rather than left open.

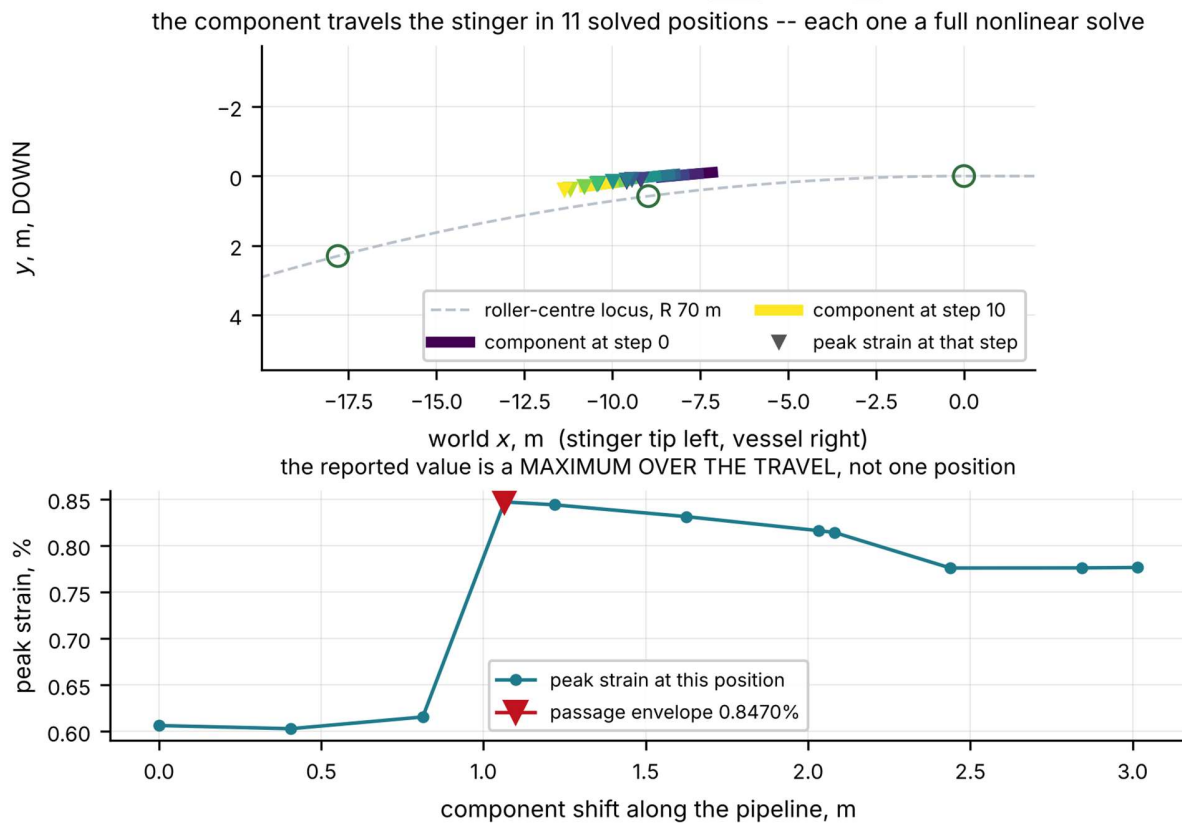


Figure 5. The passage sweep. Above, zoomed onto the travelled span: one component drawn at each of the eleven solved positions of a traverse, each a full nonlinear solve, with the worst element at each position marked. Below: the peak strain against the component's shift along the pipeline. The reported value is the maximum over the travel, not any one position.

6. Benchmark validation results

The benchmark comparison is the main technical evidence of the paper. Results are organised by the published table families so that every comparison remains traceable to its source.

6.1 The shape of the result

Across the 57 published comparisons the agreement splits by **quantity**, not by family.

	Published comparisons	Median $ \Delta $	Within 10%
Bending moment	23	2.3%	91%
Peak or region strain	57	14.1%	39%

This is the paper's main engineering finding and it is visible in every family that publishes both quantities. Three mechanisms are consistent with it. The first is a property of the quantities themselves and is quantified below; the other two are candidates, recorded as such rather than resolved:

- **past yield, a strain is a far more sensitive quantity than the moment that produced it**
- the reported value is an **envelope over a traverse** where the published value is a single position,
- the Tool reports **extreme-fiber strain** where the source studies report a nominal strain at integration points.

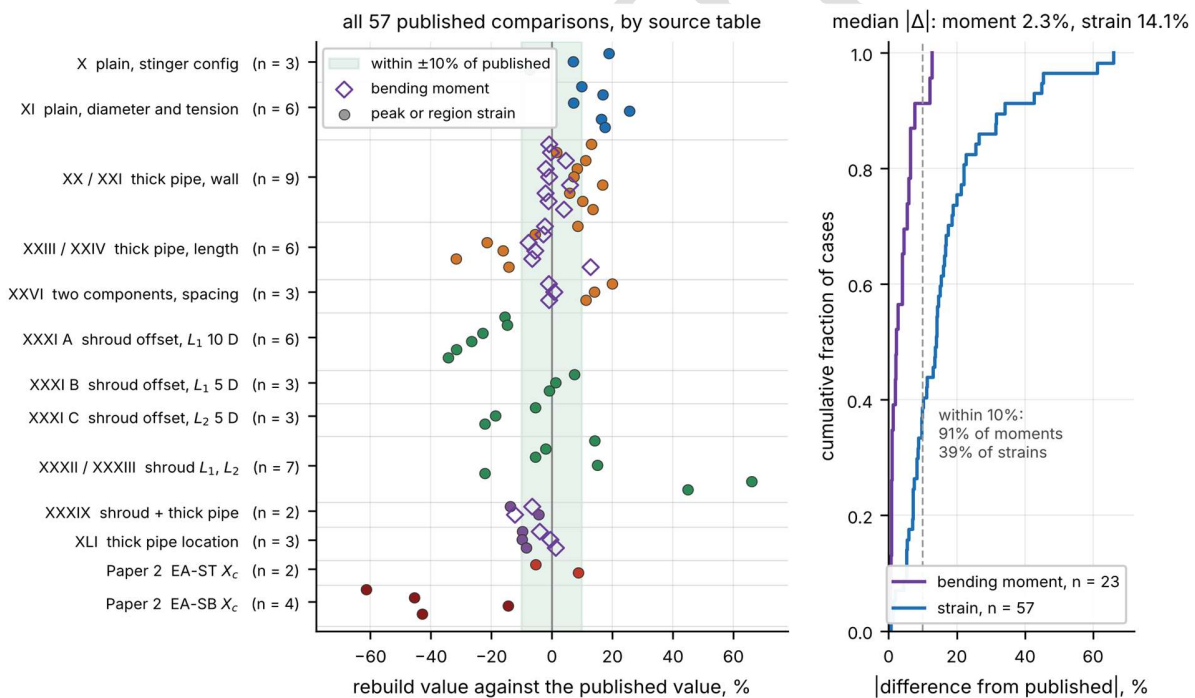


Figure 6. All 57 published comparisons. Left: each case as a distance from its published value, grouped by source table and coloured by family; the band is $\pm 10\%$. Right: the same as cumulative distributions. The separation between the moment curve and the strain curve is the finding.

Table 4 — Benchmark results overview

Benchmark family	Published reference	Current result	Interpretation
Plain pipe, stinger radius	Study 1, TABLE X	Peak strain +18.9%, +7.1%, -7.3%	Trend reproduced; the curves cross between 85 and 105 m
Plain pipe, diameter and tension	Study 1, TABLE XI	Analytical $D/2R$ reproduced to <1%; excess over it roughly twice the published FEA	Analytical anchor exposes local strain amplification
Short thick pipe	Study 1, TABLE XX/XXI	Moment -2.1% to +6.0%; strain +1.6% to +16.8%	Global mechanics far stronger than local strain
Long thick pipe	Study 1, TABLE XXIII/XXIV	Moment -7.7% to +12.8%; strain -31.6% to +8.6%	Multi-roller engagement reproduced; published saturation is not
Two thick components	Study 1, TABLE XXVI	Moment within 1%; strain +11.3% to +20.0%	Moment excellent; the spacing effect is 3.7× too strong
Shroud offset depth	Study 1, TABLE XXXI/XXXIV	Monotonic rise reproduced in all three configurations; -34.2% to +7.5%	Main trend reproduced, rate not
Shroud length and taper	Study 1, TABLE XXXII/XXXIII	All seven passages complete; -22.1% to +66.1%	An earlier invalid comparison corrected
Shroud with thick pipe	Study 1, TABLE XXXIX/XLI	Strain -4.3% to -13.7%; moment -12.2% to +1.3%	Worst-case position reproduced
EA-ST	Study 2, F2	X_c -5.3% and +8.8%	Good connector-region agreement
EA-SB	Study 2, F1 and F2	X_c -14.4% to -61.3%	The main unresolved family

6.3 Plain pipe

Plain pipe cases check the basic overbend response before any inline component is added. They also include the only analytical reference in the benchmark set.

For the 16 in pipe at 120 MT, the current tool differs from the published strain values by +18.9%, +7.1% and -7.3% at $R = 70$ m, 85 m and 105 m. The radius trend is reproduced, with best agreement at 85 m.

At zero tension, the response can be checked against:

$$\varepsilon = D / 2R$$

The tool estimate matches this analytical value within 1% for all three diameters. However, it predicts a larger FEA-to-analytical strain increase than the published study: +18.8%, +21.8% and +34.7%, compared with +8%, +14% and +17%.

So the trend is correct, but the local strain concentration at rollers is higher than the benchmark. This is one of the key remaining items to investigate.

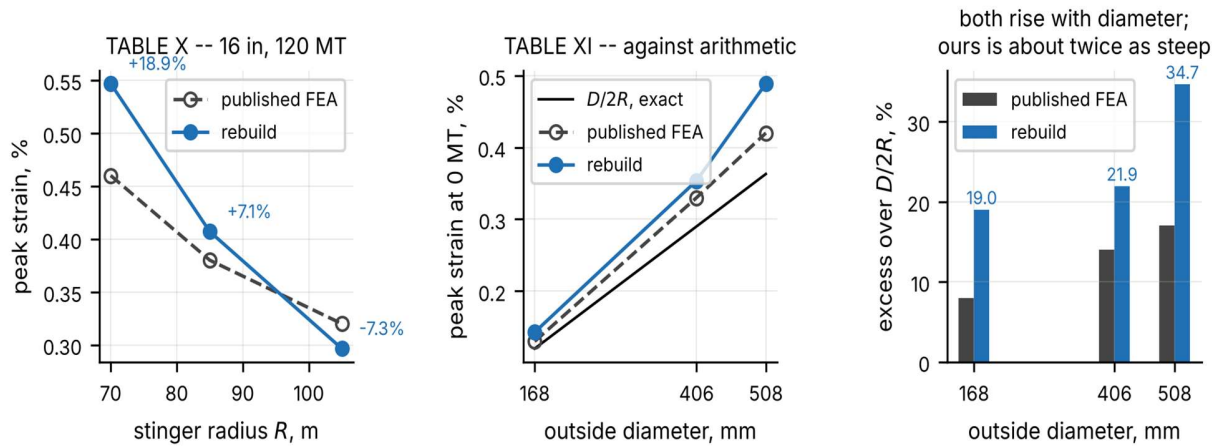


Figure 7. Plain pipe. Left: peak strain against stinger radius, TABLE X. Centre: peak strain at zero tension against the exact $D/2R$ line, TABLE XI — the Tool's analytical column reproduces the published one to under 1%. Right: the excess of each FEA over pure bending. Both rise with diameter; the Tool's rate is about twice the published one.

Table 5 — Plain pipe, detailed comparison. Reference Study 1 TABLE X (16 in, 120 MT) and TABLE XI (three diameters at $R = 70$ m, wall 21 mm). Bending moment is unpublished for both tables, so the Tool moments are predictions.

Table	Case	R, m	T, MT	Benchmark strain, %	Calculated Strain, %	Δ
X	A	70	120	0.460	0.5468	+18.9%
X	B	85	120	0.380	0.4070	+7.1%
X	C	105	120	0.320	0.2965	-7.3%
XI	6in-T0	70	0	0.130	0.1428	+9.9%
XI	6in-T100	70	100	0.290	0.3391	+16.9%
XI	16in-T0	70	0	0.330	0.3536	+7.2%
XI	16in-T100	70	100	0.420	0.5280	+25.7%
XI	20in-T0	70	0	0.420	0.4888	+16.4%
XI	20in-T100	70	100	0.540	0.6352	+17.6%

6.4 Thickened pipe components

Thick-pipe cases check the effect of a stiffened span in the overbend region.

Moment agreement is strong. Across nine wall-thickness cases, component-body moment differs by -2.1% to +6.0%, with eight cases within 5%. Strain agreement is weaker, at +1.6% to +16.8%. The two-component spacing cases show the same pattern: moment is within 1%, while strain differs by +11.3% to +20.0%.

The main open issue is component length. The published study shows strain saturation beyond about 20D, but the tool does not fully reproduce this. At $R = 85$ m and 20D length, the strain difference is -31.6%, while moment remains within -7.7% to +12.8%.

The long cases also exposed a contact defect. Initially, long component body nodes could not bear on rollers because contact was selected by ownership label. After switching to structural line contact, the model reproduced multi-roller engagement. Figure 10 shows the corrected behaviour.

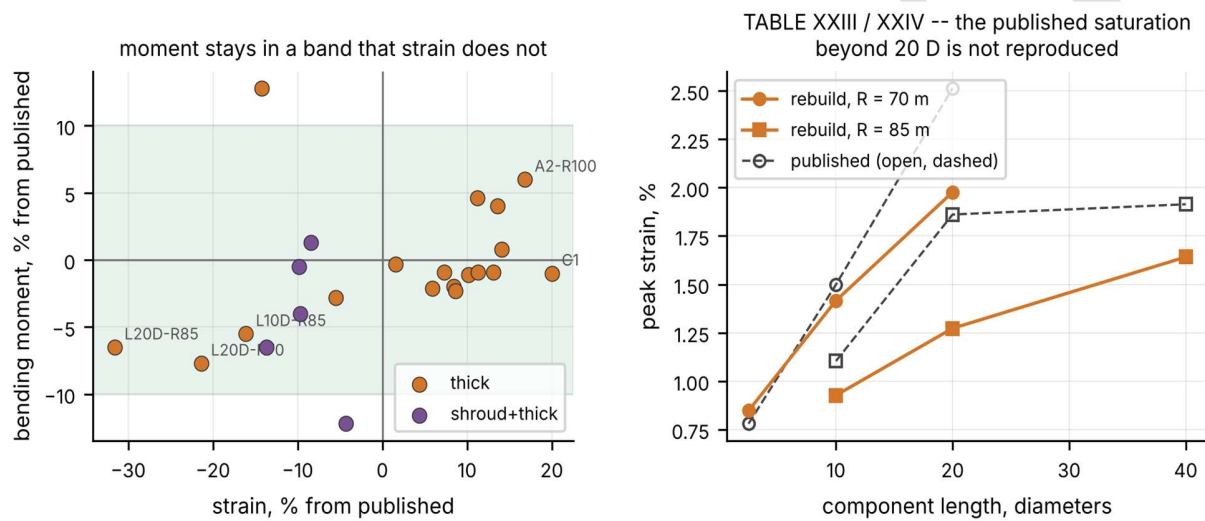


Figure 8. Thick-pipe components. Left: every case for which both a moment and a strain are published, plotted against each other. The moment stays in a band the strain does not — this is Section 6.0's finding in one panel. Right: peak strain against component length; the published saturation beyond 20 D is not reproduced.

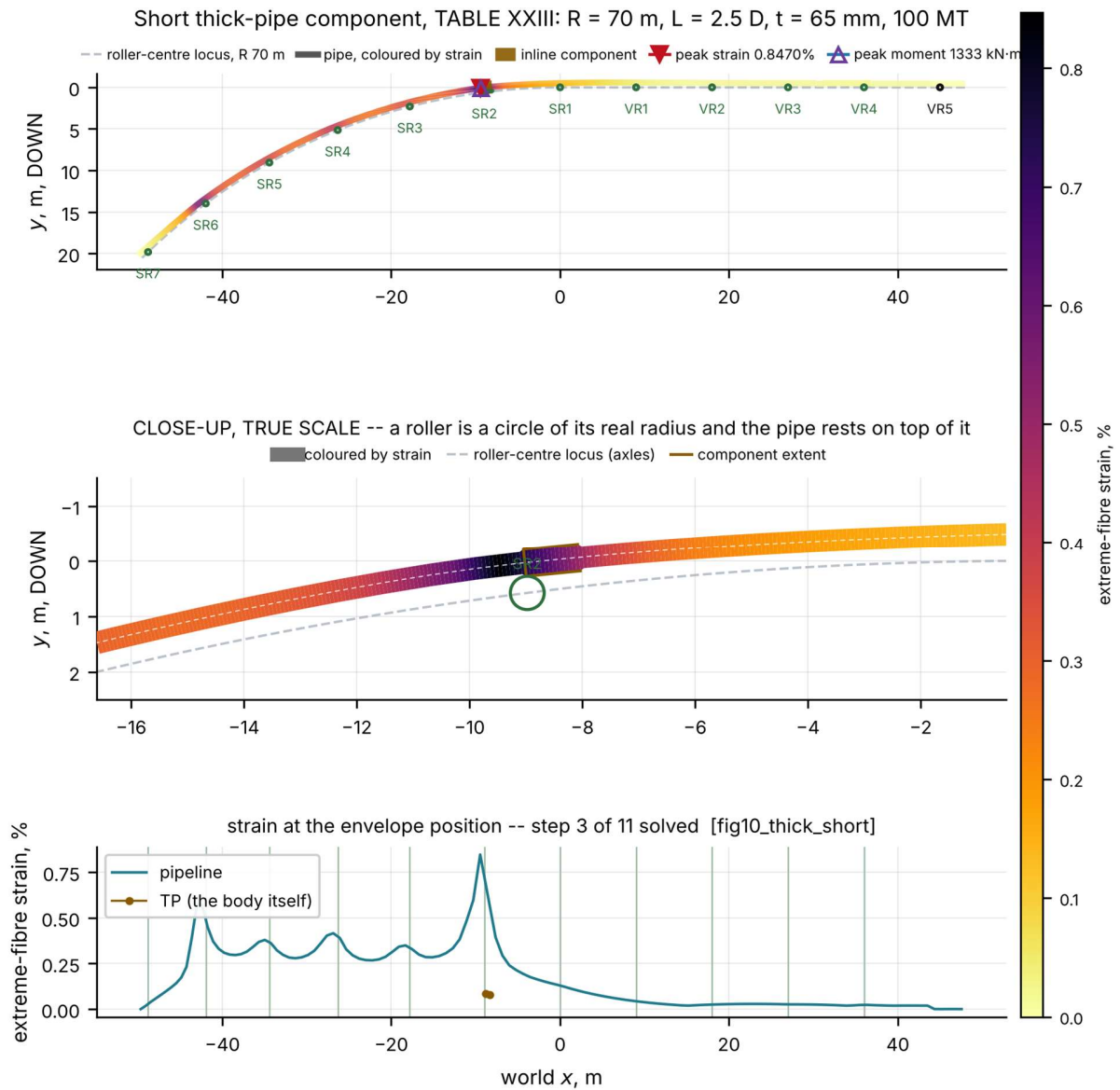


Figure 9. A short thick-pipe component, TABLE XXIII at $R = 70\text{ m}$, $L = 2.5\text{ D}$, $t = 65\text{ mm}$, 100 MT . Top: the deformed pipe at the envelope position, with the stinger arc, the roller stations, the component body and the worst element in strain and in moment. Bottom: extreme-fibre strain along the pipeline. The component itself is barely strained; the peak is in the pipe immediately beside it. Artifact fig10_thick_short; the envelope, 0.8470% , reproduces the ledger's 0.8471% .

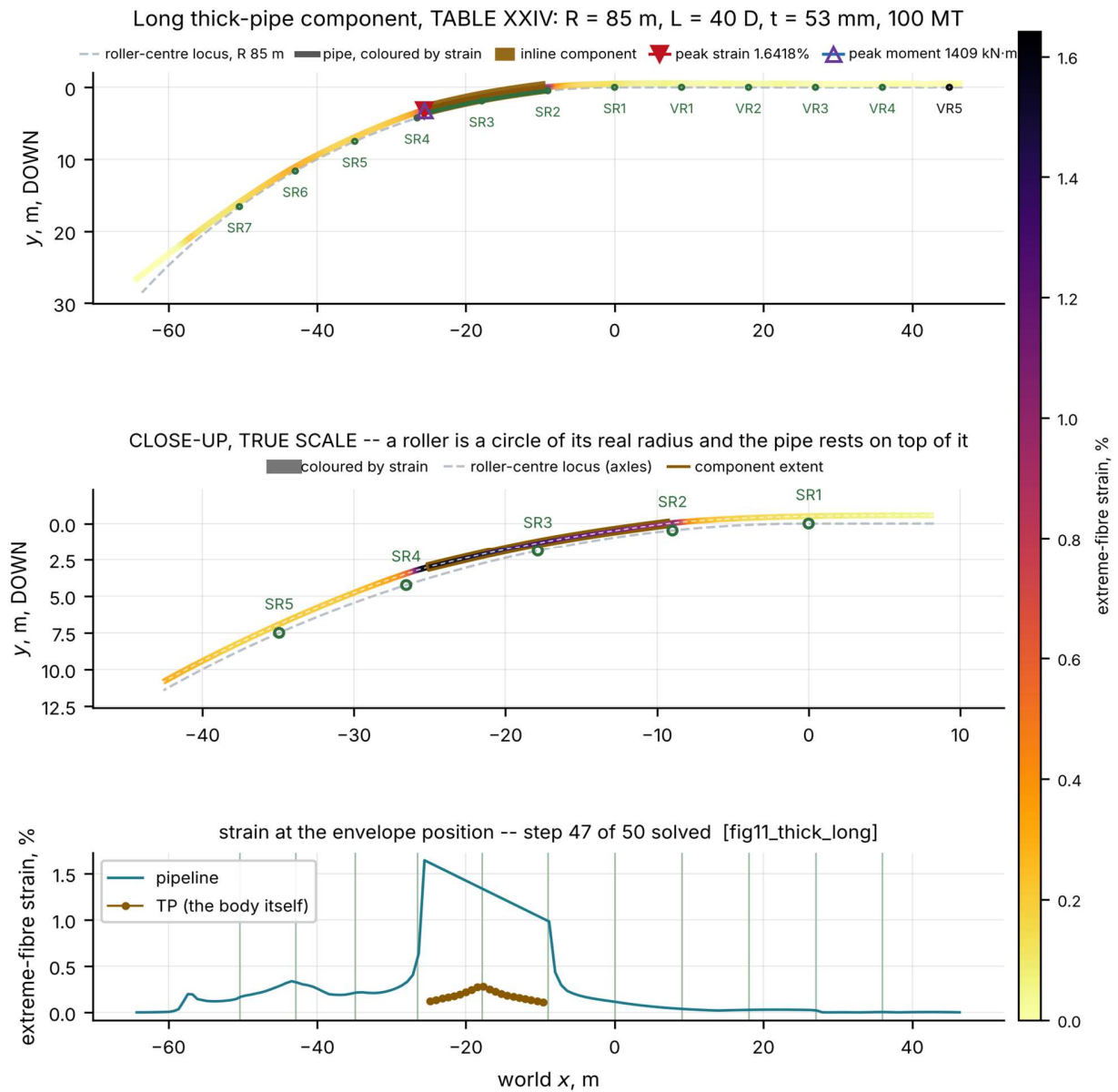


Figure 10. A long thick-pipe component, TABLE XXIV at $R = 85$ m, $L = 40$ D, $t = 53$ mm, 100 MT. The component now spans three roller stations at once — the multi-roller engagement that the contact-chain defect used to prevent. Artifact fig11_thick_long; the envelope, 1.6418%, reproduces the ledger's 1.6413%.

Table 6 — Thick-pipe components, detailed comparison. Reference Study 1 TABLE XX/XXI (wall thickness, 2.5 D component), TABLE XXIII/XXIV (component length) and TABLE XXVI (two components at three spacings). Moment is read on the component body, which is where the published “peak moment” is reported.

Source Table [4]	Case	Varied	Benchmark strain, %	Calculated Strain, %	Δ	Benchmark moment, kN·m	Calculated Moment, kN·m	Δ
XX-XXI	A1-R70	t _{mm} = 32	0.562	0.6358	+13.1%	1311	1299.2	-0.9%
XX-XXI	A1-R85	t _{mm} = 32	0.473	0.4804	+1.6%	1251	1247.1	-0.3%
XX-XXI	A1-R100	t _{mm} = 32	0.339	0.3771	+11.2%	1131	1182.7	+4.6%
XX-XXI	A2-R70	t _{mm} = 42	0.647	0.7015	+8.4%	1347	1320.6	-2.0%
XX-XXI	A2-R85	t _{mm} = 42	0.508	0.5452	+7.3%	1288	1276.1	-0.9%
XX-XXI	A2-R100	t _{mm} = 42	0.358	0.4183	+16.8%	1144	1212.7	+6.0%
XX-XXI	A3-R70	t _{mm} = 53	0.727	0.7701	+5.9%	1366	1337.5	-2.1%
XX-XXI	A3-R85	t _{mm} = 53	0.556	0.6127	+10.2%	1311	1297.0	-1.1%
XX-XXI	A3-R100	t _{mm} = 53	0.395	0.4487	+13.6%	1184	1230.8	+4.0%
XXIII-XXIV	L2.5D-R70	L _{OD} = 2.5	0.780	0.8471	+8.6%	1384	1352.0	-2.3%
XXIII-XXIV	L10D-R70	L _{OD} = 10	1.499	1.4161	-5.5%	1650	1604.0	-2.8%
XXIII-XXIV	L20D-R70	L _{OD} = 20	2.514	1.9763	-21.4%	2223	2053.0	-7.7%
XXIII-XXIV	L10D-R85	L _{OD} = 10	1.104	0.9267	-16.1%	1581	1494.0	-5.5%
XXIII-XXIV	L20D-R85	L _{OD} = 20	1.861	1.2728	-31.6%	1977	1848.0	-6.5%
XXIII-XXIV	L40D-R85	L _{OD} = 40	1.914	1.6413	-14.2%	2883	3253.0	+12.8%
XXVI	C1	spacing_OD = 2.5	0.771	0.9253	+20.0%	1384	1369.8	-1.0%
XXVI	C2	spacing_OD = 10	0.760	0.8671	+14.1%	1371	1382.4	+0.8%
XXVI	C3	spacing_OD = 20	0.751	0.8362	+11.3%	1365	1352.1	-0.9%

6.5 Shroud components

Shroud cases check contact lift without a matching increase in pipe bending stiffness. The pipe sits inside an offset body, which holds it away from the roller line. Strain is reported by the source study's regions: X1 to X5.

Where published moment values are available, agreement is reasonable. Only five shroud cases have published bending moment. For these, the calculated strains differs by -12.2% to +1.3%, with four of five within 7%. Strain differences for the same runs are -4.3% to -13.7%. For the remaining shroud cases, calculated moment values are reported as predictions because no published moment is available.

The main trend is reproduced: deep-section strain increases with offset depth. However, the rate is not fully matched. In the shortest, high-tension configuration, agreement is within 7.5%. In the longest configuration, the published strain increases faster with offset depth than the Tool, so the difference grows from -15.5% at 0.75D to -34.2% at 3.00D.

The length and taper cases exposed an important workflow issue. Five of seven passages had previously been compared with published results before completing their full travel. After adding completion status and travel cutback, all seven passages completed. The long-shroud dual-roller trend then appeared in the correct direction, although the magnitude remained weaker than the published result: -10.7% compared with -52%. One completed case also exceeded the material table range, which led to adding the material-strain guard.

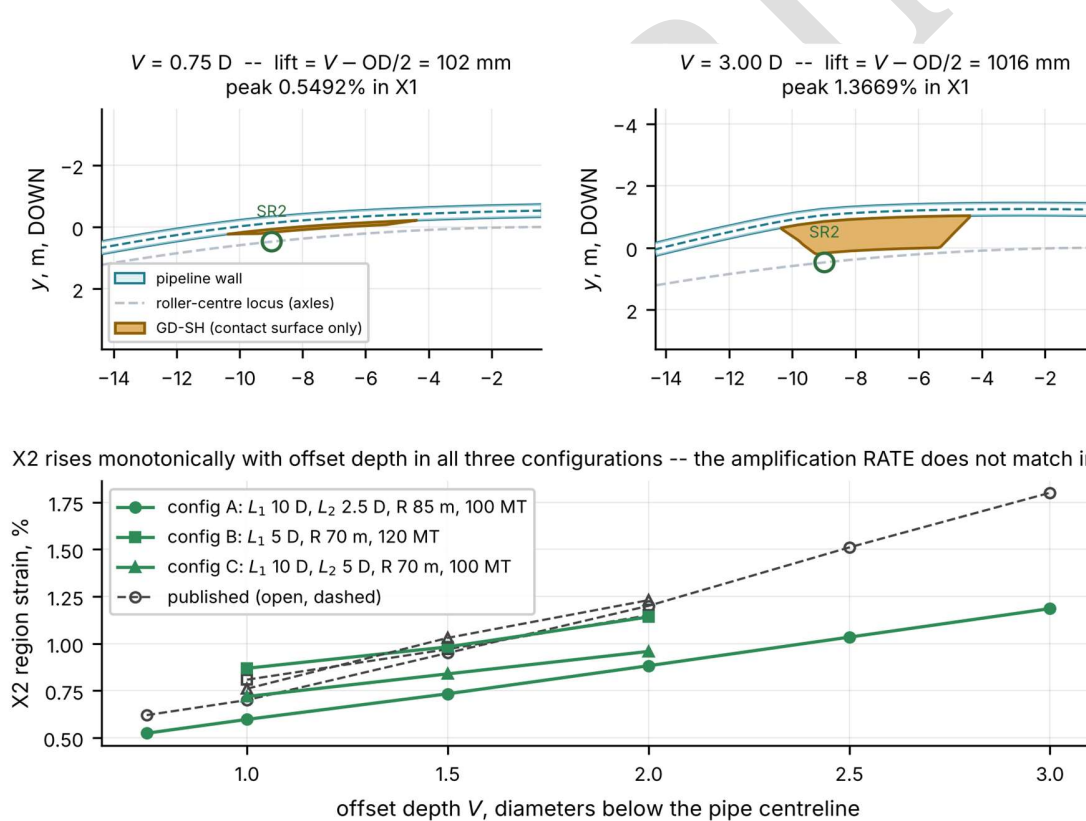


Figure 11. Shroud offset depth. Top: the deformed pipe near the offset body at $V = 0.75 D$ and $V = 3.00 D$, zoomed, with the gap between the roller line and the pipe centreline shaded — that gap is the mechanism. Bottom: deep-section strain against offset depth in all three published configurations. The rise is reproduced everywhere; the acceleration in configuration A is not.

Table 7 — Shroud and shroud-plus-thick-pipe, detailed comparison. Reference Study 1 TABLE XXXI/XXXIV (offset depth in three configurations), TABLE XXXII/XXXIII (deep-section and taper length) and TABLE XXXIX/XLI (a thick pipe inside the shroud). Strain is the X2 deep-section region except where the note says the peak fell elsewhere. **The source study publishes a bending moment only for TABLE XXXIX and XLI;** the Tool moment is given for every case, as a prediction where there is nothing to compare it against.

Table	Case	Varied	B'mark strain, %	Calc' Strain %	Δ	B'mark moment (kNm)	Calc' Strain (kNm)	Δ
XXXI-A	A-V0.75	P_v_OD = 0.75	0.620	0.5239	-15.5%	—	1247.5	—
XXXI-A	A-V1.00	P_v_OD = 1	0.700	0.5973	-14.7%	—	1274.0	—
XXXI-A	A-V1.50	P_v_OD = 1.5	0.950	0.7332	-22.8%	—	1303.3	—
XXXI-A	A-V2.00	P_v_OD = 2	1.200	0.8820	-26.5%	—	1329.0	—
XXXI-A	A-V2.50	P_v_OD = 2.5	1.510	1.0338	-31.5%	—	1351.6	—
XXXI-A	A-V3.00	P_v_OD = 3	1.800	1.1852	-34.2%	—	1368.9	—
XXXI-B	B-V1.00	P_v_OD = 1	0.808	0.8685	+7.5%	—	1332.9	—
XXXI-B	B-V1.50	P_v_OD = 1.5	0.970	0.9822	+1.3%	—	1358.0	—
XXXI-B	B-V2.00	P_v_OD = 2	1.150	1.1410	-0.8%	—	1372.3	—
XXXI-C	C-V1.00	P_v_OD = 1	0.760	0.7187	-5.4%	—	1317.6	—
XXXI-C	C-V1.50	P_v_OD = 1.5	1.030	0.8389	-18.6%	—	1335.0	—
XXXI-C	C-V2.00	P_v_OD = 2	1.230	0.9580	-22.1%	—	1352.6	—
XXXII-XXXIII	S2-1	L1_OD = 2.5	0.780	0.8911	+14.2%	—	1327.1	—
XXXII-XXXIII	S2-2	L1_OD = 4	0.750	0.7353	-2.0%	—	1317.5	—
XXXII-XXXIII	S2-3	L1_OD = 10	0.760	0.7187	-5.4%	—	1317.6	—
XXXII-XXXIII	S2-4	L1_OD = 10	0.790	0.9089	+15.1%	—	1345.1	—
XXXII-XXXIII	S2-6	L1_OD = 10	1.230	0.9580	-22.1%	—	1352.6	—
XXXII-XXXIII	S2-7	L1_OD = 25	0.660	1.0960	+66.1%	—	1342.4	—
XXXII-XXXIII	S2-8	L1_OD = 50	0.590	0.8553	+45.0%	—	1336.1	—
XXXIX	TP5D	L_TP_OD = 5	0.952	0.8220	-13.7%	1405	1314.1	-6.5%
XXXIX	TP10D	L_TP_OD = 10	1.260	1.2052	-4.3%	1555	1365.0	-12.2%
XLI	loc-X2	TP_location = 2	0.901	0.8132	-9.7%	1362	1307.5	-4.0%
XLI	loc-X3	TP_location = 3	0.744	0.6714	-9.8%	1310	1302.9	-0.5%
XLI	loc-X4	TP_location = 4	0.744	0.6811	-8.4%	1271	1287.5	+1.3%

6.6 Externally attached structures

Externally attached cases check connector-driven load transfer. The frame is connected to the pipe at discrete fastening points, either above the pipe as EA-ST or below the pipe as EA-SB. EA-SB also lifts the pipe away from the normal roller arc. The source paper [5], mentions several types of connections, F1, F2, PS, PSD etc. Currently this program has studied only F1 and F2 connection cases (1 or 2 fixed connections). Reason being

that this case causes the highest bending strains in pipeline. Strain is reported using the source study's regions: Xc near connectors, Xi between connectors and Xe outside the connector span.

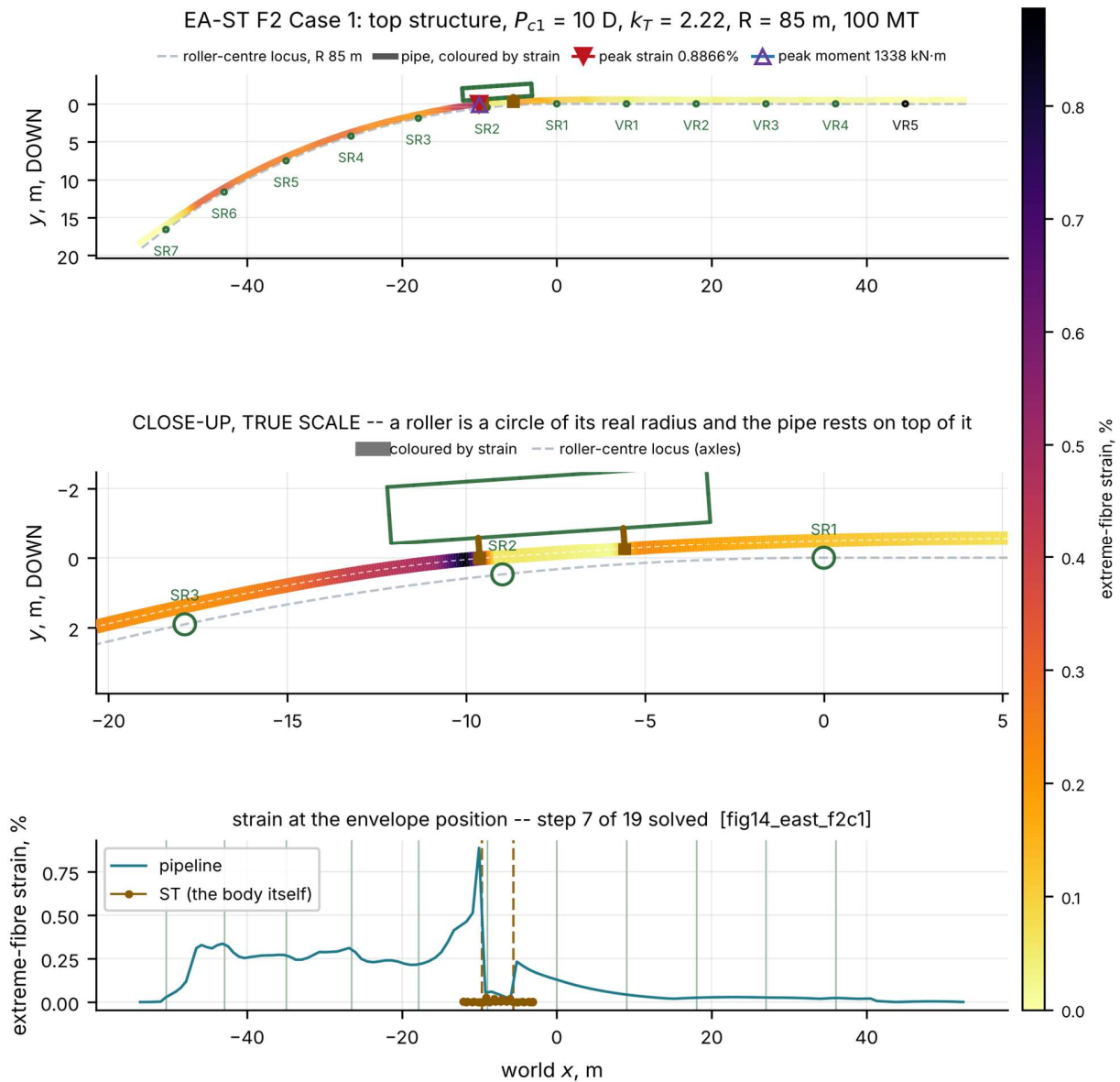


Figure 12. EA-ST F2 Case 1 — a top structure at $P_{c1} = 10 D$, $k_T = 2.22$, $R = 85$ m, 100 MT. The frame bears on the rollers and reaches the pipe only at its two connectors, which is the shielding the source study describes.

The main strain ordering is reproduced in five of the six cases. Connector regions generally have the highest strain, the interior pipe is strongly shielded, and the outboard pipe behaves closer to plain overbend. In the five cases with an interior region, Xc remains much larger than Xi, by factors of 9.2 to 17.5.

EA-ST also agrees reasonably well in magnitude. The two EA-ST cases differ from the published Xc values by -5.3% and +8.8%, both within 10%.

Peak location should be read with mesh resolution in mind. In the EA-ST case shown in Figure 12, the worst strain is reported in the element next to the connector, not exactly at the connector point. This is expected

because strain is reported at element positions, while the connector load enters at a node. The reported peak is still inside the Xc band.

EA-SB shows more deviation from the source cases when the peak strains are high. This needs further evaluation.

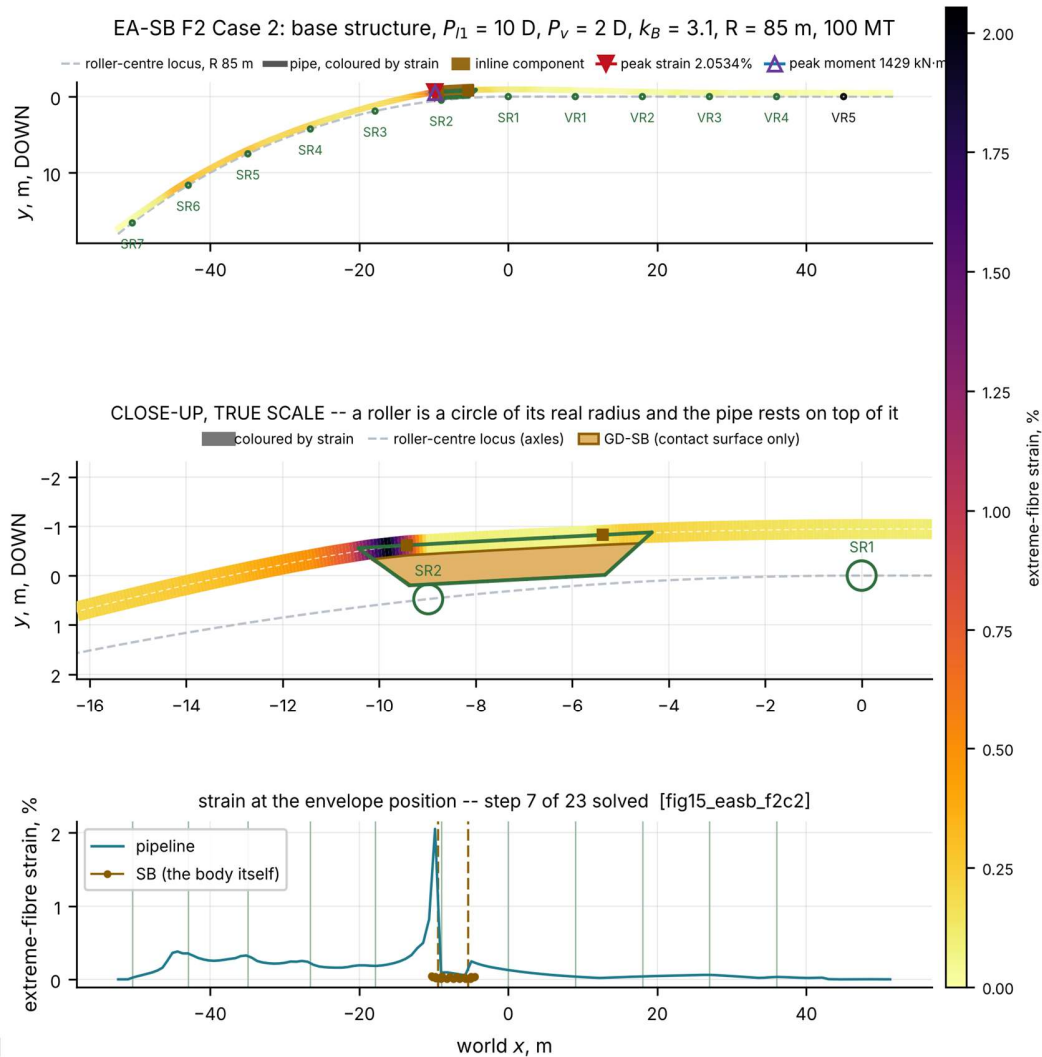


Figure 13. EA-SB F2 Case 2 — a base structure at $P_{11} = 10 D$, $P_v = 2 D$, $k_B = 3.1$, $R = 85$ m, 100 MT. The base stiffens the pipe and holds it off the arc at the same time, which is why it is mechanically unlike EA-ST. Artifact fig15_easb_f2c2; the envelope, 2.0534%, reproduces the ledger's 2.053%. This case is the best of the four EA-SB cases, at -14.4%.

Table 8 — EA-ST and EA-SB connector-region comparison. Reference Study 2, R = 85 m, 100 MT. X_c is the peak within two diameters of a connector.

Family	Case	B'mark Strain at X_c	Calc' Strain at X_c	Δ	B'mark Strain at X_e	Calc' Strain at X_e	Δ
EA-ST	F2 Case 1	0.936%	0.887%	−5.3%	0.732%	0.519%	−29.2%
EA-ST	F2 Case 2	1.410%	1.534%	+8.8%	1.021%	0.669%	−34.5%
EA-SB	F1 Case 1	2.30%	0.889%	−61.3%	1.41%	1.053%	−25.3%
EA-SB	F2 Case 1	2.24%	1.222%	−45.4%	1.45%	1.208%	−16.7%
EA-SB	F2 Case 2	2.40%	2.053%	−14.4%	1.52%	0.828%	−45.5%
EA-SB	F2 Case 3	2.53%	1.447%	−42.8%	1.60%	1.321%	−17.5%

F1 Case 1 is the only row in which the Tool's X_e exceeds its X_c , inverting the published ordering.

7. What the benchmarks exposed

The benchmark campaign did more than generate comparison numbers. It exposed many defects in the application-layer physics and workflow. **None was in the solver kernel. None was found by a test, a type checker or a code review. They were found by a published number disagreeing with a computed one,** and in some of them, the computed one had converged, completed and printed without complaint.

Table 9 — Defects exposed by benchmark validation

Defect	Effect	Correction	Lesson
Contact selected by owner label	Inline component body nodes could not bear on rollers, so a long component never engaged more than one	Select the contact chain by structural line, not by reporting label	A reporting label is not a physics label
Truncated passage compared to a complete published value	Five of seven shroud cases were tabulated against the published result having run 61.6% to 93.7% of their travel; nothing in the artifact said so	Completion status written into every artifact, and travel cut back between positions	Every result row needs completion evidence, carried on the row
Lay tension fixed to a material point	As the pipe moved through the passage the load stayed bolted to one piece of steel, leaving a free cantilever of travel length with 100 MT on its tip	Apply lay tension at a station, along that station's tangent	A moving-boundary load needs its coordinate system stated explicitly

These defects support the paper's central point. Application-level engineering software can fail even when the underlying solver is sound. The risk is not only a mathematical formulation error; it can be a contact-chain definition, a load-coordinate interpretation, an incomplete passage or a parameter nobody transcribed. LLMs can help find and fix such issues quickly, but only if the workflow forces them to appear through benchmark evidence.

8. Using the program to generate machine-learning training data

A validated analysis program is useful for more than answering one case at a time. It is a **data source**, and the shortage of data is the reason machine learning has made less progress in installation engineering than the method's general success would suggest.

8.1 Why the data has to be generated

Overbend response is a strongly nonlinear function of several interacting parameters — stinger radius, roller spacing, pipe diameter, wall thickness, lay tension, and whatever inline structure is passing — and a designer at concept stage wants an answer in seconds, not in the hour a full nonlinear passage takes. That is a surrogate-modelling problem, and it needs training data spanning the design space rather than clustered at the few points a project happened to analyse.

Manually produced FEA data does not span it. Building a commercial-FEA model of one configuration is a day of skilled work; varying it is more. The consequence is familiar: a published study reports tens of cases chosen to make a point, every one of them modelled by hand, and each is valuable precisely because it was expensive. Both reference studies behind this paper are of that kind — between them they publish 50 cases. Fifty cases is enough to validate a program and far too few to fit a surrogate over five interacting parameters.

A validated program removes the cost and keeps the physics. Once the case definition is a set of numbers and the runner is a command, the marginal cost of a case is machine time: the passages behind Section 6 run in seconds to minutes each, so a run of over a thousand is an overnight job on a few cores rather than a year of modelling. What the data then inherits is the program's validation — its agreement, and its disagreements, as recorded in Section 6 — rather than anyone's confidence in it.

This is also where the LLM-assisted workflow pays off a second time. The parameter sampler, the sharded runner, the result schema, the merge step and the checks are all exactly the sort of well-specified, programming-heavy task Section 10.2 identifies as the strength of the approach, and they are what stands between a solver that can answer a case and a program that can produce a dataset.

9. Discussion

9.1 Main engineering finding

An existing benchmarked nonlinear beam solver can be adapted into a practical S-lay overbend analysis program, and the quality of the result is not uniform across the quantities it reports. **Global bending moment is reproduced to a median 2.3% and qualitative trends are reproduced almost everywhere. Local peak strain is reproduced to a median 14.1% and is where the remaining work is.** That split is consistent across four component families.

9.2 Main LLM finding

The LLM was most useful where the task was well specified but programming-heavy: building modules, refactoring scripts, adding runners, preparing validation ledgers, creating tests and maintaining documentation. It was less suitable as an independent judge of physics. The domain expert still had to decide what the program should represent and what evidence counted as validation.

The defect record of Section 7 sharpens that. **Five defects, none in the solver, none found by a test, all five found by a published number disagreeing with a computed one.** Three of them produced results that converged, completed and printed without complaint, which is the specific failure mode an LLM-assisted workflow should be expected to produce: code that is syntactically clean, internally consistent, well tested against its own assumptions, and answering a slightly different question than the one asked. A test suite written by the same agent that wrote the code tests the same misunderstanding twice.

There is a second observation, less comfortable. The single most consequential correction in the campaign required the source paper to be read, and the agent did not have it. For most of this project's life the two reference studies were not in the repository at all — only their case tables had been transcribed — and no control in Section 4.2 could have detected it. The boundary of what the workflow can verify is the boundary of what has been put inside it.

9.3 Generalization beyond S-lay

The broader lesson is that benchmarked solvers can be customised for other engineering situations using a similar workflow: lifting and rigging checks, installation-aid assessment, simplified structural screening, pipeline free spans, frame idealisation, concept-stage component assessment. The solver provides the mathematical core; the domain expert defines the physical idealisation; the LLM implements the application layer and its supporting software; the published benchmark decides whether the result may be used.

This matters for engineering organisations. Many useful internal tools do not require a new solver from first principles. They require careful adaptation of existing validated methods into repeatable workflows. LLMs can reduce the programming barrier for that adaptation, **provided the benchmark gate exists before the first line of application code is written** — because, as Section 7 shows, it is the gate and not the code review that finds what is wrong.

10. Conclusions

This paper reframes LLM-assisted engineering software development as solver *customization* rather than solver *invention*. The nonlinear beam solver and the plasticity capability were established in earlier benchmarked work. The contribution here is the adaptation of that kernel into a practical S-lay overbend program for pipelines and inline components, and the account of what it took to make the result believable.

The tool reproduces important benchmark behaviours: plain-pipe radius trends, the analytical pure-bending limit to under 1%, short thick-pipe component moments to -2.1% to $+6.0\%$, multi-roller engagement after a contact-chain correction, the monotonic rise of shroud strain with offset depth in all three published configurations, and top-structure connector-region response within 10%. Across all 57 published comparisons the moment agrees to a median 2.3% and the strain to a median 14.1%

It also shows unresolved differences: local strain amplification roughly twice the published rate against an analytical reference, no reproduction of the published saturation of strain with component length, and base-structure connector-region strain low by up to 61%. These open items strengthen rather than weaken the methodological point. **Benchmark comparison is what prevents LLM-assisted programming from becoming unverified automation.**

The same program is also a data source, and that is the second reason to build one this way. Manually modelled FEA is expensive enough that the published literature on this problem is measured in tens of cases, which will not support a surrogate over the parameters that matter. A validated program makes the marginal cost of a case machine time, and inherits its credibility from the benchmark record rather than from anyone's confidence in it — provided it records, per case, the provenance, the mesh, the derived features and the convergence, completion and validity status that make a row safe to train on. A plain-pipelay pilot of about 1500 cases has been initiated on that basis; it is reported separately.

The broader conclusion is that domain experts can use LLMs to customise existing benchmarked solvers into practical engineering tools with much less direct programming burden. The expert does not need to implement every software detail personally, but must still define the physics, select the benchmarks, audit the inputs and judge the results. In this model the LLM is a powerful programming layer around engineering judgement, not a replacement for it.

References

1. S. Sivaraman and J. Venkataramana Reddy, "Toward AI-Assisted Conceptual Design of Subsea Inline Structures: Classification and Strain Behaviour of Inline Components during S-Lay Installation," *International Journal for Research in Applied Science and Engineering Technology (IJRASET)*, vol. 14, no. VII, pp. 1127–, July 2026. doi:10.22214/ijraset.2026.84268.
2. S. Sivaraman and J. Venkataramana Reddy, "Advancing AI-Assisted Conceptual Design of Subsea Inline Structures: Mechanical Behaviour of Pipeline-Mounted Structural Assemblies during S-Lay Installation," *International Journal for Research in Applied Science and Engineering Technology (IJRASET)*, vol. 14, no. VIII, pp. 394–, August 2026. doi:10.22214/ijraset.2026.84577.

3. S. Sivaraman and J. V. Reddy, "Knowledge-Graph-Governed AI for Engineering Design: A Case Study in S-Lay Inline Structures," engrXiv preprint 8245, 16 September 2026. <https://engrxiv.org/preprint/view/8245>.
4. S. Sivaraman, "AI-Assisted Development of a Nonlinear Finite Element Analysis Program: A Case Study Using a Large Language Model," International Journal for Research in Applied Science & Engineering Technology (IJRASET), vol. 14, no. VI, June 2026. doi: 10.22214/ijraset.2026.83397.
5. S. Sivaraman, "AI-Assisted Extension of a Nonlinear FEA Program to Material Plasticity: Benchmark-Driven Validation against Commercial FEA," International Journal for Research in Applied Science & Engineering Technology (IJRASET), vol. 14, no. VII, July 2026. doi: 10.22214/ijraset.2026.84267.