**Duchlaufträger Applikation**

Version 1.0 Oktober 2023

Die vorliegende Version behebt einige gravierende Probleme der vorhergehenden Versionen, die im Wesentlichen durch 2 Entscheidungen verursacht waren.

1. Die Speicherung der geometrischen Informationen (Balkenkoordinaten) erfolgt zur Sicherung der Konsistenz des Rechenmodells ausschließlich bei den Übertragungspunkten (Redundanz) und nicht bei den Balkenelementen wie Lasten und Lager.   
   Das Ziel einer interaktiven und intuitiven Benutzerumgebung erfordert jedoch die interaktive Veränderungsmöglichkeit aller Balkenelemente. Diese müssen in ihrer Beschreibung und Lage flexibel verändert werden können. Das wiederum erfordert, dass die zugehörigen Übertragungspunkte konsistent editiert und gelöscht werden können und auch neue hinzugefügt werden. Das Lösungsverfahren erfordert auch eine Ordnung der Übertragungspunkte in aufsteigender Form.   
   Diese Anforderungen haben die Umsetzung des Lösungsverfahrens für alle möglichen Konstellationen von Balkenelementen (Punktlasten, Linienlasten, Lager), deren Lage, gegenseitige Beeinflussung und deren konsistente Zuordnung zu Übertragungspunkten relativ kompliziert und fehleranfällig gemacht. Wird z.B. eine Linienlast interaktiv gelöscht, so muss überprüft werden, ob die zugehörigen Übertragungspunkts auch gelöscht werden müssen oder aber von anderen Balkenelementen noch benötigt werden.
2. Der *Beschreibung* der Momentenverläufe ist exakt bekannt und durch quadratische Polynomfunktionen definiert. Die *Darstellung* der Momentenverläufe kann in C#, .NET nicht exakt visualisiert werden, da Polynomfunktionen für die geometrische Beschreibung und Visualisierung nicht zur Verfügung stehen. Dies gilt auch für die meisten anderen Programmierumgebungen. Folglich mussten Momentenverläufe für die Darstellung approximiert und interpoliert werden. Unbeeinflusst davon sind die analytischen Berechnungsergebnisse, die als Zahlenwerte dargestellt werden.  
   Eine sehr elegante und mächtige Beschreibung funktionaler Verläufe steht mit den sogenannten Bezier-Splines zur Verfügung. Diese wurden in vorhergehenden Versionen sowohl in einfacher wie auch in quadratischer Form verwendet. Das hat sich im Nachhinein als sehr problematisch herausgestellt. Der wesentliche Grund dafür liegt nach meiner Erfahrung begründet in der Tatsache, dass wesentliche Bestandteile der funktionalen Beschreibung - die sogenannten Kontrollpunkte - NICHT AUF der Kurve selber liegen, sondern teilweise weit entfernt davon. Das hat sich vor allem bei Skalierungsvorgängen, aber auch bei ingenieurtechnischen Beurteilungen wie Maxima und Nulldurchgänge als problematisch herausgestellt.   
   Daher wurde die Darstellung in dieser Version umgestellt auf lineare Polygonzüge. Das sieht zwar im ersten Schritt nach einem „brute force“ Vorgehen aus, hat sich aber für die gegebenen Anforderungen als absolut angemessen, sehr leistungsfähig und vor allem auch in der Programmierumgebung leicht umsetzbar herausgestellt. Dies ist natürlich auch nur eine Approximation und Interpolation, aber sämtliche Kurvenpunkte stimmen mit dem funktionalen Verlauf exakt überein und liegen AUF der Kurvendarstellung. Ein kleiner Nachteil ist dabei, dass Kurvenmaxima und Nulldurchgänge nicht unbedingt exakt erfasst werden. Dies hat sich aber bei einigermaßen angemessener Auflösung der Kurveninterpolation als unproblematisch herausgestellt. Die Berechnung und Anzeige von Kurvenmaxima und Nulldurchgängen erfolgt auf der Basis der Kurveninterpolation (nicht auf Basis der analytischen Lösung), da dies einfach und flexibel für alle möglichen Konstellationen allgemein umsetzbar war.

**Erstes Fazit der Umsetzung einer Ingenieurapplikation in C#, .NET**

Aus Sicht der Informatik ergibt sich mit den Möglichkeiten der Anwendungsentwicklung in dieser Entwicklungsumgebung grundsätzlich nichts Neues an Möglichkeiten und Funktionalitäten. Alles, was hier implementiert wurde, ließ sich auch schon in der „alten“ Entwicklungsumgebung von *C++, Windows Forms* umsetzen. Allerdings gestehen auch Informatiker vielfach den resultierenden Ergebnisse zu, dass diese oft Code produzieren, der kunstvoll erstellt wurde, schwierig bis kaum nachvollziehbar ist und sich einem Dritten nur sehr schwierig erschließt. Diese Tatsache hat sogar in kommerziellen, sogenannten „legacy“ Anwendungen dazu geführt, dass bestimmte Code-Bestandteile nach Jahren ihrer Erstellung nicht mehr angetastet werden. So ist z.B. von dem ursprünglichen, alten AutoCAD bekannt, dass hocheffiziente, funktionierende, zentrale Bestandteile der Kernimplementierung nicht mehr angetastet wurden, da ihre Umsetzung so kunstvoll und kryptisch war, dass sie den aktuellen Entwicklern nicht mehr bekannt war und sich ihnen auch nicht mehr erschlossen hat („never touch a running system“).

Aus Sicht der Ingenieure ist dies nicht akzeptabel. Computercode für Ingenieuranwendungen muss so gestaltet sein, dass Lösungsalgorithmen problemlos erkennbar und nachvollziehbar sind und dass Erweiterungen und neue Verfahren als Ergebnisse der Forschung von Ingenieuren transparent implementiert und in bestehende Lösungen integriert werden können. Dabei ist es jedoch unverzichtbar und muss unbedingt gewährleistet bleiben, dass moderne Ingenieuranwendungen an den Ergebnissen der Forschung und Entwicklung in der Informatik partizipieren und an aktuellen Ergebnissen teilhaben.

Der schiere Umfang an Daten in modernen Ingenieuranwendungen, die transient im Kernspeicher moderner Rechner oder persistent in externen Speichermedien gespeichert und verwaltet werden, war vor wenigen Jahren noch unvorstellbar und verbietet es Ingenieuren mit alt herkömmlichen Verfahren auf diese Datenmengen zuzugreifen und diese zu verwalten. Frühe Implementierungen der FE-Methode haben z.B. die Speicherverwaltung noch selber mit Integer-Pointern in einem Blank-Common Block in FORTRAN implementiert. Dies ist heute unvorstellbar.

Allein das „Aufräumen“ der Speicher nach dem Ablauf komplexer Programmsystem („garbage collection“) ist heute eine anspruchsvolle Aufgabe, die verhindern soll, dass der verfügbare Speicherplatz scheinbar kontinuierlich abnimmt („memory leakage“). Aktuelle Ingenieurapplikationen sind meist unzureichend geschützt vor diesen Problemen.

Die allgemeine und selbstverständliche Nutzung mehrerer, hochkomplexer, paralleler Rechenkerne sogar in marktüblichen Laptops geht heute weit über die Vorstellungen eines linearen Programmablaufs hinaus und muss entsprechend organisiert und synchronisiert werden.

Die Anforderungen an die Erscheinungsform und Funktionalitäten der Benutzeroberflächen moderner Programmsysteme (UI) werden heute selbstverständlich und in standardisierter Form gefordert.

Aus meiner Sicht ist es die Aufgabe der Informatik, diese Aufgabenstellungen anwendungsübergreifend zu lösen und die Aufgabe der Ingenieure, die aktuellen Ergebnisse der Forschung und Entwicklung in der Informatik für moderne Ingenieuranwendungen zu nutzen.

Ebenso bedarf es aus meiner Sicht für diesen Prozess immer noch einen Transformationsmechanismus, der die Übertragung der Ergebnisse der Informatik in die Anwendungsdisziplinen leistet und anwendbar macht. Im Bauingenieurwesen ist dies die Bauinformatik, die sicherstellen muss, dass die aktuellen Entwicklungen und Konzepte der Informatik in Anwendungsentwicklungen im Bauingenieurwesen umgesetzt und genutzt werden können.

Die Informatik wird und kann in der Regel auch keine Forschung und Entwicklung von Bauingenieurapplikationen übernehmen.

Der allgemeine Bauingenieur wird neue Lösungsverfahren in seinem Fachgebiet entwickeln und bestehende weiterentwickeln. Eine herkömmliche „Programmierung“ dieser Ergebnisse ist heute den Anforderungen nicht mehr angemessen und kann kaum moderne Entwicklungen und Forschungsergebnisse der Informatik umsetzen und nutzen.

Die Umsetzung neuer Lösungsverfahren in moderne Ingenieurapplikationen muss von spezialisierten Fachleuten auf beiden Gebieten, der Informatik und der Anwendung, erfolgen, Eine reine Auftragsvergabe an Spezialisten wird den heutigen Anforderungen nicht mehr gerecht und eine kooperative Entwicklung von Fachleuten verschiedener Disziplinen gestaltet sich fast immer sehr schwierig.

Im Prinzip verschiebt sich die Grenze zwischen Systemsoftware und Anwendungsprogrammierung seit Jahren kontinuierlich. War es anfangs noch erstrebenswert „open software, open source“ für Betriebssysteme zu fordern und zu nutzen, ist dies schon seit einigen Jahren in vielen Anwendungsgebieten weder sinnvoll noch möglich. Ähnlich gestaltet sich die Entwicklung zwischen Betriebssystem und Anwendungsprogrammierung. Haben die Anwendungsprogrammierer anfangs noch Konzepte bis zur Programmierung auf Pozessorebene (Assembler) genutzt, ist dies heute weder sinnvoll noch nötig. Heute stehen extrem leistungsfähige Compiler zur Verfügung, die weit über die Möglichkeiten normaler Programmierer für Ingenieurapplikationen hinaus gehen. Inzwischen hat sich die die Grenze bis zur Nutzung einer sogenannten „Middleware“ verschoben, die viele moderne Konzepte der Informatik auf der Ebene der Anwendungsprogrammierung in einer modernen Entwicklungsumgebung anbietet. Damit hat sich die Grenze zwischen geschlossener Systemsoftware und offener Anwendungssoftware wieder ein Stück weit verschoben.

Die Kehrseite sind bei dieser Entwicklung die wachsenden Abhängigkeiten von Herstellern entsprechender Systeme - proprietäre Systeme, die eigentumsrechtlich einem bestimmten Anbieter gehören und in ihren inneren Zusammenhängen verborgen und nicht nachvollziehbar sind.

Die EU ist in dieser Entwicklung einen eigenen Weg gegangen. Dort wurde akzeptiert, dass die Software geschlossen und eigentumsrechtlich geschützt ist, aber es wurde die Forderung durchgesetzt, dass die **Anwenderdaten**, die mit diesen Systemen erzeugt werden offen sein müssen. Dies erfolgt z.B. im lesbaren Datenformat XML. Die Datenformate *docx* und *xlsx* der Firma Microsoft sind hierfür 2 Beispiele. Diese konnten nur mit hohem politischen Druck durchgesetzt werden. Andererseits waren die Argumente gegen digitale Versionen rechtlich verbindlicher Dokumente, die nur in einem intransparenten, eigentumsrechtlich geschützten Datenformat einer bestimmten Firma verfügbar waren, unwiderlegbar und inakzeptabel.