REVIEW

3(8), February 1, 2013

ISSN 2278–5469 EISSN 2278–5450

Discovery

Process modeling and related algorithms analysis about soil collapse in channel construction

Feng An☼

Research Scientist, Beijing Aero-Space Control Device Department, P.R.China

☼Correspondence to: Feng An, 24th Building, 5th Li, Yv Hai Garden, North Cai Shi Road, Hai Dian District, Beijing, P.R.China, ZipCode 100143, E-Mail: fengan@iupui.edu

Publication History
Received: 19 December 2012
Accepted: 25 January 2013
Published: 1 February 2013

Citation
Feng An. Process modeling and related algorithms analysis about soil collapse in channel construction. *Discovery*, 2013, 3(8), 29-31

General Note
Article is recommended to print as color digital version in recycled paper.

ABSTRACT

In contemporary world, Channel Construction & Mineral Mining have put channel formation analysis into a very important point, The Software Dynaform can format many kinds of channels. The Author proposes a different kind of Dynaform implementation in which the whole system can be realized in a single dynaform project. And the desired work will be first donated to OPENGL to do mathematical modeling using Finite Element Analysis with 50,000 points in every single square inch. Later the system can be returned back to OPENGL where several types of deformation models can be applied specifically. Each will be sent back to OPENGL to do the additional math modeling. Finally, a maple software was written to calculate the deformation degree and estimates how to consolidate the whole tunnel channel. The modified structure was later verified using OPENGL in order to show the effectiveness of the method.

Keywords: C#, Cave-type Modeling.

Abbreviation: FEA - Finite Element Analysis

1. INTRODUCTION

Nowadays, subways' and underground channels' construction takes up a very large proportion in Industrial Engineering Projects. The Author proposes a improved analysis about tons of soil and rocks collision when performing channel construction. In this research paper, FEA (Finite Element Analysis) is applied in this topic. When performing Soil-Layer and Rock-Layer Analysis, the key importance is in deciding how to do the simulation and how to decide the FEA methods' parameters. Basics about FEA methods are listed:

1. Finite Elements must be uncontinuous, thus there must be values that are odd enough, however, since there must have gaps in between to sample values, the finite elements cannot be 100% representing the true values of the whole system design. That is to say, if the system characteristics are linear and foreseeable in this Results value range, the discrete analysis method must be OK. However, if the system is non-linear in the input X-Axis value range, the system's output cannot be identified in value, thus abrupt values can occasionally show up. This is highly dangerous when performing a construction, in order to figure those values out and try to fix the whole project draft. Performing a finite element analysis combined with a throughput band-pass field

prevention design is enough.

2. Considering the 1)'s understanding, the system must be constructed into a linear model when applying the X-Axis values into the Design Patterns, a band-pass and high-ban filter will be applied too. Considering the Dynaform Software, the tools do not possess filtering functionaries, thus, the author has found some other ways to do this problem.
3. Typical Design Patterns allow not only IDE-type software, such as Solidworks, Dynaform etc. to take up the majority of the whole system simulation, but those patterns also allow pre- or post- C compilers' modeling. Since C compilers can only simulate software models, C# is implemented to handle the problem.

2. MULTI-LAYERS ANALYSIS AND SOIL FORMATION ANALYSIS

The Design Patterns do not allow different types of soil (Multi-layers Concept) to be completely mixed when conducting the experimenting, so different types of soil may have totally different check-out data class and may results in different attitudes towards them. Thus, the compiler needs some additional information to input and at the same time can help to take scrutimize at the input FEA data. The Compiler is designed according to computational intelligence algorithms. The key ideas of it are as follows:

1. Computational Intelligence is a little different from Machine Learning, the key differences is that computational intelligence is based on history-back and the history is a certain given period of time back, so the algorithm constantly changes the history it references. So the results can change unexpectedly and can usually fit the real experimental environment.
2. The Compiler can be designed using the following way: A typical C to Assembly Correlation Algorithm for this types of implementation is C-Leyapu-Assembly Algorithm. This can be built up according to modern control theories.

The C language can build up the input/output ports of the software model. For each ports, according to Leyapu theory in Complex System Design, it must do not have poles and axis bypass in order to prevent dead-lock status to happen. Thus, the ports should be bi-directional. However, according to Complex Lumped System Design in Control theory, the system should be linear and unbounded at the same time. So that the C language model should help to maintain this by implementing a C# block, in which the C# coding should have zero threading and zero tasking, just waiting for the Input Sequence and Output Sequence to come in and out.

```
/*********************************************************************/
//Create a Sample Model for the Software Ports to Connect with the C
language protocol
FOR i = 0; i ++; i -- ; i ## % maintain the i value of the ports
        port A ## %NEED ADDITION PROGRAMMING: port A <= 100
        port B ## %NEED ADDITION PROGRAMMING: port B <= 100
        port C ## %NEED ADDITION PROGRAMMING: port C <= 100
        port D ## %NEED ADDITION PROGRAMMING: port D <= 100
        ……
END % this can help to have four samples PORT A, PORT B, PORT C, PORT D.
// to connect with the C language protocol and leave inner connection ports
with additional input parameters
FOR i = A; i –, i ++.?C# % try to connect and guarantee the C# ports are bi-
directional
                        i ++.
FOR C ++.?A ++?C –.\C# % try to implement four sample ports in whole
        PORT A ## %NEED ADDITION PROGRAMMING: port A <= 100
        PORT B ## %NEED ADDITION PROGRAMMING: port B <= 100
        PORT C ## %NEED ADDITION PROGRAMMING: port C <= 100
        PORT D ## %NEED ADDITION PROGRAMMING: port D <= 100
END
// Implement the C# sample block for each Port
FOR C# PP?Go?C# %try to build up the C# block
                INPUT
                        CASS    % relevant to PORT A
                                PARAMETERS ONE
                                        method 1.

        IMPLEMENTING C# MODULE SIX
                                        method 2.

        NULL
                        CASS    % relevant to PORT B
                                PARAMETERS TWO
                                        method 3.
                                                NULL
                                        method 4.

        IMPLEMENTING C# MODULE FIVE
                        CASS    % relevant to PORT C
                                PARAMETERS THREE
                                        method 5.
                                                NULL
                                        method 6.
                                                NULL
                        CASS    % relevant to PORT D
                                PARAMETERS FOUR
                                        method 7.
                                                NULL
                                        method 8.

IMPLEMENTING C# MODULE SIX
        END
        OUTPUT
                        CASS    % relevant to PORT D
                                PARAMETERS FIVE
                                        method 9.
                                                NULL
                                        method 10.
                                                NULL
                        CASS    % relevant to PORT E
                                PARAMETERS SIX
                                        method 11.
                                                NULL
                                        method 12.

                                                NULL
                        CASS    % relevant to PORT F
                                PARAMETERS SEVEN
                                        method 13.
                                                NULL
                                        method 14.
                                                NULL
        END
        PARAMETER CODING
                        ARS     % FOR IMPUT
                                        method 15.
                                                CODING
FIVE
                                        method 16.
                                                CODING
SIX
                        ARD     % FOR OUAPUT
                                        method 17.
                                                METHOD
FIVE IMPLEMENTATION
                                        method 18.
                                                METHOD
SIX IMPLEMENTATION
                        ARE     % DUE CPUT
```

```
                                                    method 19.
                                                            METHOD
SIX SOFTWARE*
                                                    method 20.
                                                            NULL
                        END
END
% IMPLEMENTATION OF METHOD FIVE AND SIX
METHOD FIVE
        SIX
                THREE & FOUR
        SEVEN
                EIGHT & RASPDAURE
        EIGHT
                DRASC & RSQRD
        FIVE
                ADSGH || RSQCR
        THREE
                        null.
END
METHOD SIX
        FIVE
                METHOD FIVE & CCSEAQWFGHYUICA
        THREE
                METHOD FIVE || DWSETYUIFGHJBVCXSWQ
        QU
                null.
        ARE
                null.
        CMD
                        WINDOWS XP SP3 || LINUX FEDORA CORE 2.258 &&
OPENBSD CCSRA
        QEMU
                VIRTUAL BOX 2.293
        GCC
                GCC || G++ || VIRTUALRFCASKD
                CONTINUE...
END
/*************************************************************************/
In order to provide direct correlation between two computer-coding systems
and link the higher level Computer Coding language to machine coding and
Windows Operating Systems, the system should then use another C#
program to do the job.
/*************************************************************************/
*CCS 6.0 presents
*WRITTEN BY Feng Anderson An. He has implemented a C# program to do
        *operating system's desktop show-up and server terminal show-
up functionaries
% Coding blocks: ONE
THE DESIGN BLOCKS PATTERNS
        ONE.
                C LANGUAGE COMPILER RRD
        TWO,THREE.
                C++ LANGUAGE COMPILER G++
        FOUR.
                INTEL ASSEMBLY RESOURCE DDK
        R?SKD
                RRS && CCS && DEV CPP || RRS
END
/*************************************************************************/
```

The Compiler design is not basically quite a good one since it do not have any VoIP technology in the source code, currently IPv6 protocols. This protocol is used to do data transferring with our any loss in data quality while at the same time maintain the maximum amount of lossless data. The DCT & IDCT algorithm and Bi-QUAD algorithm or even the 8-FFT BUTTFLY algorithm do not cater to this Internet Protocol Standards, however have the same types of doings. The proposed compilers do not have any input and output data yet. On deciding which types of data transmission protocol is OK for this application. The Author finally can decide which type of data is OK by performing a DSP Hardware Implementation Capability Survey. Since the DSP chip cannot implement IPv6 series protocols on board because of its loss of hardware models correlated and merely sample data samples listed in the handbook of the Internet Protocol Regulation, the Author chooses DCT and IDCT algorithms, and finally make the data transferring mechanism into a butterfly mechanism. This greatly helps the FEA Ultra High Speed (8 core CPU full-speed, $(2.8GHz)^8$ in total throughput and SPARC 2000 testing).