사용자 편의성을 고려한 연속체계 모의실험 언어의 개발[1)]

Development of a User-friendly continuous-system Simulation Language

민경하*, 임창관**, 박찬모*
K.H-Min, C.K-Yim, C.M-Park

Abstract

기존의 모의 실험언어를 이용해서 연속 체계를 모의 실험하는 것은 사용자가 언어에서 요구하는 형태로 모델을 형성해야 하는 어려움이 따른다. 따라서 본 연구에서는 사용자에게 최대한 편의성을 제공하는 연속체계 모의 실험 언어인 PCSL(Postech Continuous-system Simulation Language)를 개발하였다.

PCSL은 주어진 대상을 모델링한 미분방정식과 그것을 푸는데 필요한 여러 가지 제약 사항으로 이루어진 간단한 프로그램을 입력으로 받아 자동으로 모의 실험을 수행함으로서 사용자의 노력이 최소화되게 된다.

PCSL처리 시스템의 구성은 주어진 모델을 C 프로그램으로 변형하는 변환기, 모의 실험 알고리즘을 구현한 C 프로그램을 생성하는 생성기, 모의 실험을 수행하는 실행기, 사용자 인터페이스 등으로 되어 있다.

구현 예로는 먼저 선형 상미분방정식의 예로 mass-damper-spring system, 비선형 상미분방정식의 예로 van der Pol 방정식, 연립 상미분방정식의 예로는 mixing tank problem 등을 보였다.

1. 서론

컴퓨터를 이용한 모의실험 방법은 과학 및 공학 분야뿐만 아니라 사회 과학 분야의 문제를 푸는데에도 유용한 도구이다. 따라서 컴퓨터를 이용한 모의실험 언어는 대상이 되는 분야의 복잡한 현상과 법칙등을 모델링하고 원하는 결과를 얻는데 필요한 프로그램 작성에 편리하여야 한다. 그러나 기존의 컴퓨터 모의실험 언어들을 사용하기 위해서는 컴퓨터에 대한 전문 지식과 프로그래밍 실력이 요구된다. 그러므로 일반인이 사용하기에 편리한 언어와 사용자 인터페이스를 개발하는 것은 중요한 일이다.

컴퓨터를 이용한 모의실험중에서도 연속체계 모의실험

1) 본 연구는 한국과학 재단과 포항공대 정보통신연구소가 부분적으로 지원하였음을 밝힙니다.

* 포항공과대학 전자계산학과
** 삼성전자

은 모델의 대부분이 미분방정식의 형태로 만들어진다[1]. 그런데 해석학적으로 해답을 구할 수 있는 미분방정식은 그 형태가 극히 제한되어 있다. 따라서 미분방정식의 해를 수치해석학적으로 구하는 방법이 연구되어왔고, 또 그러한 기능을 갖는 모의실험 언어가 많이 발표되었다. 이러한 모의실험 언어를 이용해서 미분방정식을 풀기 위해서는 주어진 미분방정식을 모의실험 언어가 요구되는 형태로 변환시켜야 한다. 예를 들면, 모의실험 언어인 MIMIC을 이용해서 n차 상미분방정식의 해를 구하기 위해서는 주어진 미분방정식을 n개의 1차 상미분방정식으로 변환해야 한다[2].

이 연구에서는 사용자에게 최대한의 편의를 제공하는 연속체계 모의실험 언어인 PCSL(Postech Continuous-system Simulation Language)을 개발하였으며 사용자는 모델을 미분방정식으로 표현만 하면 필요한 변환과 실행 프로그램을 자동으로 발생하도록 하였다.

2. 연구 방향

2.1 기존의 모의 실험 언어에 대한 고찰

2.1.1 ACSL

ACSL(Advanced Continuous Simulation Language)은 적당한 크기의 현장에서 발생하는 문제를 모의 실험하기 위해서 설계되었다. *ACSL*에서는 입력으로 모델을 정의한 CSL 파일과 매개변수값의 지정이나 필요한 run-time 명령어들을 저장한 DAT 파일을 받아들인다. 그리고 ACSL 전처리기에서 입력된 *ACSL* 프로그램을 FORTRAN으로 변환한 다음, *ACSL run-time library*와 링크해서 기계어 코드를 생성한다. 그리고 생성된 기계어 코드를 실행하는 동안에 ACSL은 사용자가 매개변수의 값을 바꿀 수 있는 *interactive mode*로 변환된다. 이러한 2단계 컴파일은 기계에 대한 독립성을 얻을 수 있고, 또 대부분의 컴퓨터에서 FORTRAN 컴파일러를 제공하기 때문에 모의실험 언어에서 널리 쓰이는 기법이다[5].

2.1.2 DARE-P

*DARE-P*는 *ACSL*보다 먼저 개발된 CDC용의 연속체계 모의실험 언어이다. ACSL과 비슷한 기능을 갖지만, 그 문법이 간단해서 배우기가 쉽기 때문에 교육용으로 쓰인다. CDC가 완전한 ACSII 코드를 갖지 않은 이유로, 대부분의 *DARE-P*는 소문자를 지원하지 않는다[8].

2.1.3 DESIRE

*DESIRE*는 최고의 수행 속도와 최대한의 대화성을 갖춘 연속 체계 모의실험 언어이다. *DESIRE*는 2단계 컴파일을 하지 않는 소수의 언어인데, 그 이유는 실행 속도를 빠르게 하기 위한 것이다. 즉, 전처리기가 입력된 프로그램을 다른 목적어(target language)로의 변환을 하지 않고, 따라서 이식성(portability)이 낮다. 유연성(flexibility), 대화성(interactivity), 반응성(responsibility) 등에서 탁월한 반면 프로그램이 쉽지 않고, 신뢰도가 낮으며, 견고성(rotubstness)이 부족하다[11].

2.2 PCSL의 개요

PCSL은 미리 정의된 문법에 따라서 입력된 code를 처리해서 연속체계에 대한 모의실험을 수행한다. 입력되는 code 중에서 가장 중요한 것은 미분방정식을 나타내는 부분으로, 그 형태는 곱의 합 꼴인 함축형(implicit form)으로 제한한다. 이렇게 제한하는 이유는 곱의 합 꼴인 함축형이 미분방정식 중에서 가장 일반적인 형태이고, 또 처리하는 과정에서 고려해야 할 다른 사항들을 제거함으로써, 발생할지도 모르는 오차를 줄이기 위한 것이다.

PCSL에서 미분방정식을 푸는 방법은 먼저 입력된 미분방정식을 변형해서 프로그램으로 바꾸고, 미분방정식의 해를 수치해석학적으로 구할 수 있는 C 프로그램을 자동 생성한 다음, 미분방정식의 변형된 형태와 결합시키고 그 프로그램을 수행해서 결과를 얻는 것이다[3,4,6].

2.3 PCSL 처리시스템의 구성

PCSL 처리시스템의 구성은 미분방정식을 해석해서 digital-analog simulation(DAS)기법으로 풀 수 있는 형태로 변환하는 번역기, 이렇게 변환된 형태의 미분방정식과 프로그램상의 여러 가지 제약 사항들을 고려해서 C 프로그램을 생성해 주는 생성기, 생성된 C 프로그램을 실행시켜

서 그 결과를 얻는 실행기, 그리고 사용자에게 편리한 입출력 방법을 제공하는 사용자 인터페이스로 되어있다.

PCSL 처리 시스템이 PCSL 프로그램을 처리하는 과정은 다음과 같다.

1. 먼저 번역기가 입력된 모델을 해석하고, 주어진 미분방정식을 C 프로그램의 형태로 바꾼 f.c를 생성한다.
2. 생성기에서는 모델에 대한 모의 실험을 하는 DAS 알고리즘을 구현한 C 프로그램인 main.c를 생성한다.
3. 실행기에서는 실행에 필요한 정보들을 PCSL 프로그램에서 추출해서 main.h라는 파일을 만든 다음, 위에서 생성된 각 프로그램들을 실행해서 결과를 얻는다.
4. 사용자 인터페이스는 프로그램의 입력을 쉽게 하기 위한 그래픽 사용자 인터페이스의 형태로 구현된 도구(tool)와 출력 결과를 사용자가 쉽게 이해하게 하는 도구로 구성된다.

2.4 PCSL의 문법

PCSL에서 필요로하는 사항들은 미분방정식, 상수, 독립변수의 범위, 초기값, 출력할 변수 등이다. 이러한 사항들을 효과적으로 입력받기 위하여 그림 1과 같은 형태의 문법을 정의한다.

```
# This is a syntax of PCSL program.

BEGIN
VARIABLE
    INDEP :
    DEP :
    SUB :
CONSTANT
    const1 :
    const2 :
    const3 :
RANGE
    R_BEGIN :
    R_END :
    R_STEP :
INITIALS
    init1 :
    init2 :
MODEL
    Eqn1
    Eqn2
SUBMODEL
    SubEqn1
    SubEqn2
END
```

〈그림 1〉 문법

예를 들어, 제철 공정 중, 전로에서 철을 생산할 경우에, 쓰이는 탄소의 변화량이 다음의 식으로 주어졌다면, 그 경우의 PCSL 프로그램은 그림 2와 같다[10].

$$\frac{dC}{dt} = \frac{-1200}{1600} \cdot \frac{RK_{O\rho HM}}{W_{HM}} \cdot \frac{P_T}{P_{CO}+2P_{CO_2}} \cdot \left(O - \frac{P_{CO}}{f_c C f_o K_{CO} G^2}\right)$$

```
# This program is simulation program for Carbon

BEGIN
VARIABLE
    INDEP : t
    DEP : C
    SUB : O
CONSTANT
    RK_0 : 48.0
    Rho_HM : 7.0
    f_c : 0.583
    f_o : 1.0
    G : 1.25
    W_HM : 345.0
    P_CO : 0.9
    P_CO2 : 0.1
    P_T : 0.909
    K_CO : 528.51
RANGE
    R_BEGIN : 0
    R_END : 20
    R_STEP : 1
INITIALS
    C_INIT_0 : 4.18
MODEL
    "Cprm(1) = -(1201/1600)*(RK_0*Rho_HM/W_HM)*P
                T*(O-P_CO/(f_c*C*f_o*K_C"O*G*G))"
```

```
SUBMODEL
    "O = 0.04[0..1]"
    "O = 124.4066132/345[1..5]"
    "O = 104.7634638/34"
END
```

〈그림 2〉 탄소의 변화량을 모의 실험하는 PCSL 프로그램

3. 번역기의 구현

PCSL 처리 시스템의 번역기는 입력된 프로그램 중에서 미분방정식으로 표현된 모델을 C 프로그램으로 변환하는 역할을 한다. C 프로그램으로의 변환을 위해서 번역기는 입력되는 모델의 종류를 결정한 다음 각각의 종류에 대해서 서로 다른 변환 알고리즘을 적용한다. 변환의 결과로 만들어지는 파일이 f.c이다.

3.1 하나의 상미분방정식의 변환

본 절에서는 n차의 상미분방정식을 변환해서 푸는 방법을 제시한다. n차의 상미분방정식을 풀기 위해서는 주어진 방정식을 n개의 연립 1차 상미분방정식으로 변환시켜야 한다. 그러기 위해서는 먼저 f( )=0의 함축형으로 주어진 미분방정식을 최고 미분 차수에 대하여 푼 명시형(explicit form)으로 바꾸어야 한다. 그리고 변환된 명시형에서 y(i)를 y_i라는 변수로 치환하는 과정에서 n개의 연립 1차 미분방정식을 얻어야 한다. 이 장에서는 명시형으로 바꾸는 과정과 연립 미분방정식을 유도하는 과정을 설명하고 그 알고리즘을 제시한다.

3.1.1 명시형으로의 변환

주어진 n차 미분방정식을 명시형으로 바꾸는 과정은 주어진 미분 방정식에서 최고차항을 찾은 다음, 그 항을 제외한 다른 모든 항들을 이항시키고 최고차항의 계수로 나누는 것이다. 함축형으로 주어진 미분방정식을 명시형으로 바꾸는 알고리즘, *makeExplicit*은 그림 3과 같다.

```
MakeExplicit(expr)
{
    {Get Highest order term from expression}
    highest ← GetHighest(expr);
    {Make diff. eqn. with left terms}
    expr ← MakeLeft(expr);
    {Get coefficient from highest term}
    coef ← GetCoef(hihgest);
    if coef exists
      {Remove coefficient from highest term}
      highest ← RemCoef(highest, coef);
      {Divide left part with coeffcient}
      expr ← DivCoef(expr, coef);
```

〈그림 3〉 알고리즘 *MakeExplicit*

이 알고리즘을 미분방정식에 적용한 예는 다음과 같다.

$$(x+1)*y'' '-1*y''+A1*x*y'-A2*x*\exp(x)*\cos(x)=0$$

$$\Downarrow$$

$$y'' '=-1*(-1*y''+A1*x*y'-A2*x*\exp(x)*\cos(x))/(x+1)$$

3.1.2 n개의 연립1차 미분방정식으로의 변환

n차의 미분방정식을 풀기 위해서는 n개의 1차 미분방정식으로의 변환이 필요하다. 따라서 명시형으로 주어진 n차 미분방정식의 일반적인 형태를 고려하면 다음과 같다.

$$y^{(n)}=f(t, y^{(0)}, y^{(1)}, \cdots, y^{(n-1)}), \quad (1)$$

명시형의 n차 미분방정식을 n개의 1차 미분방정식으로 변환하기 위해서는 먼저 다음과 같은 의미를 갖는 새로운 변수 $y_j, j=0, 1, \cdots, n$을 정의해야 한다.

$$y_0 \equiv y \quad (\equiv y),$$
$$y_1 \equiv y'_0 \quad (\equiv y^{(1)}),$$
$$y_2 \equiv y'_1 \quad (\equiv y^{(2)}),$$
$$\cdot \quad \cdot \quad \cdot$$
$$\cdot \quad \cdot \quad \cdot$$
$$y_n \equiv y'_{(n-1)} (\equiv y^{(n)}),$$

이렇게 정의한 y_j, $j=0, 1, \cdots, n$을 이용하면 (1)의 식

은 다음과 같은 n개의 1차 미분방정식으로 변환된다.

$$\begin{aligned} y'_0 &= y_1, \\ y'_1 &= y_2, \\ &\cdots \\ y'_{n-1} &= y_n \end{aligned}$$

$$y_n = f(t, y_1, y_2, \cdots, y_{n-1}) \quad (2)$$

여기서 적분을 이용해서 y의 값을 구하는 식으로 바꾸면 다음과 같다.

$$\begin{aligned} y_n &= f(t, y_1, y_2, \cdots, y_{n-1}) \\ y_{n-1} &= \int y_n dt \\ &\cdots \\ y_1 &= \int y_2 dt \\ y_0 &= \int y_1 dt (\equiv y) \end{aligned}$$

이상에서 설명한 변환 과정을 수행하는 알고리즘은 그림 4에 나타나 있다.

```
Trans(expr)
{
  {Get order from expression}
  order ← GetOrder(expr);
  {Substitute y^(i) to be yi}
  for i = order−1 to 0 do
    expr ← Substitute(i, expr)
  {Set y^(order) to be expr}
  y_order = expr
  {Generati n 1st-order diff. eqn.}
  for i = order−1 to 0 do
    y1 = Integ(y_{i+1}, y_i(0))
}
```

〈그림 4〉 알고리즘 *Trans*

위에서 예로 든 상미분방정식에 대해서 이 알고리즘을 적용한 예는 다음과 같다.

$$y_2 = \text{Integ}(-1*(-1*y_2+A1*x*y_1-A2*x*\exp(x)*\cos(x))/(x+1))$$
$$y_1 = \text{Integ}(y_2)$$
$$y_0 = \text{Integ}(y_1)$$

3.2 연립 상미분방정식의 변환

PCSL에서 다루는 연립 상미분방정식은 1개의 독립변수와 n개의 종속변수에 대해서 n개의 방정식이 주어지는 경우를 가정한다. n개의 종속변수는 각각 서로 다른 최고 미분 차수를 가질 수 있으며, 미분방정식들 속에 자유롭게 분포된다. 독립변수를 x, 종속변수를 y_i, $1 \leq i \leq n$이라 하고, 각각 m_i, $1 \leq i \leq n$의 최고 미분차수를 갖는다면, n개의 연립 미분방정식은 다음과 같은 형태로 주어진다.

$$\begin{aligned} f_1 &\left(x, y_1^{(1)}, \cdots, y_1^{(m_1)}, \cdots, y_n^{(1)}, \cdots, y_n^{(m_n)}\right) = 0 \\ f_2 &\left(x, y_1^{(1)}, \cdots, y_1^{(m_1)}, \cdots, y_n^{(1)}, \cdots, y_n^{(m_n)}\right) = 0 \\ &\cdots \\ f_n &\left(x, y_1^{(1)}, \cdots, y_1^{(m_1)}, \cdots, y_n^{(1)}, \cdots, y_n^{(m_n)}\right) = 0 \end{aligned}$$

3.2.1 전처리과정

연립 상미분방정식을 풀기 위해서는 각 미분방정식에 대해서 앞 절에서 설명한 과정을 적용해야 한다. 즉, 각 미분방정식을 명시형으로 변환하고, 각각을 또 최고차를 고려한 연립 미분방정식으로 변환하는 과정을 적용해야 한다. 그러기 위해서는 먼저 종속변수들의 수와 종류, 그리고 각 종속변수들에 대해서 최고차항과 그 미분차수를 알아내야 한다. 또 어느 미분방정식에 그 최고차항들이 포함되어 있는지에 대해서도 알아내야 한다. 그 다음 과정으로 각 미분방정식에 대해서 명시형으로 만들 최고차항을 결정하는 일이 필요하다. 한 미분방정식에 있는 종속변수들 중에서 최고차항이 하나이면 큰 문제가 없으나 두 개 이상이면 그 중에서 어떤 항으로 명시형을 만들 것인지를 결정해야 한다. 이러한 과정을 쉽게 하기 위해서 주어진 미분방정식들을 처리하기 편리한 형태로 변환하고, 또 필요한 정보들을 추출하는 전처리과정이 필요하다.

전처리에서 필요한 자료 구조 입력된 미분방정식은 곱의 합꼴로 되어있기 때문에 미분방정식을 각 곱의 항으로 분해한다. 따라서 각 곱의 항에는 종속변수가 포함되어 있

는 경우와 그렇지 않은 경우로 구분된다. 각 미분방정식은 circularly linked list로 연결하고, 각 항들은 doubly linked list로 연결한다. 이에 대한 자료구조는 그림 5와 그림 6과 같다.

```
struct handler {
        int id;
        int size;
        int flag;
        struct token *root;
        struct handler *next;
}
```

〈그림 5〉 구조체 *handler*

```
struct token {
        int sign;
        int order;
        char *var;
        char *coeff;
        char *diff;
        struct token *next, *prev;
}
```

〈그림 6〉 구조체 *token*

위의 자료 구조에서 구조체 handler는 각 미분방정식을 나타내며, 구조체 token은 미분방정식의 각 항들을 나타낸다. 종속변수가 있는 항은 order가 0보다 큰 값이며, 그렇지 않은 항은 종속변수가 0보다 작은 값은 갖는다.

그리고 각 미분방정식에서 추출한 정보로 종속변수에 대한 최고 미분차항에 대한 정보와 각 미분방정식이 어떤 종속변수의 최고 미분차항을 가지고 있는지에 대한 정보를 나타내기 위한 자료구조로 struct term과 struct item을 정의한다. 그 형식은 그림 7과 8에 나타나 있다.

```
struct term {
        char *var;
        int order;
        int subs;
        char *expl;
        struct term *next;
}
```

〈그림 7〉 구조체 *term*

```
struct item {
        int id;
        int cnt;
        char *var;
        int order;
        struct item *next;
}
```

〈그림 8〉 구조체 *item*

위의 구조체 term은 종속변수의 최고 미분차항에 대한 정보를 나타낸다. 즉, 종속변수, 미분차수, 그 변수에 대해서 변환된 미분방정식의 명시형등을 포함하고 있다. 구조체 item은 각 미분방정식에 포함된 종속변수중에서 최고 미분차수인 항에 대한 정보를 나타내기 위한 것으로 미분방정식의 id, 최고 미분차수인 종속변수, 그 차수, 갯수 등에 대한 정보를 갖는다. 각각의 구조체들은 linked list로 구현되었다.

미분방정식의 전처리 알고리즘 각 미분방정식을 곱의 항으로 분해해서 구조체 handler와 token에 저장하는 알고리즘을 *make-token*이라 하고, 최고 미분차항에 대한 정보를 추출하는 알고리즘을 *make-term*, 각 미분방정식에 포함된 최고 미분차항에 대한 정보를 추출하는 알고리즘을 *make-item*이라 하면, 각각의 알고리즘은 그림 9와 그림 10, 그림 11에 나타나 있다.

```
make-token(systems of ODE)
{
  for each ODE {
    allocate handler;
    while(ODE is not empty) {
          allocate token;
          remove 1 term from ODE;
          store the term in token;
          connect the token;
    }
    connect the list of token to handler;
    connect the handler;
  }
```

```
  return the list of handler;
}
```

〈그림 9〉 알고리즘 *make-token*

```
make-term(list of handler)
{
  for each handler {
    for each token connected to the handler {
      if the variable in the token ∈ term;
        allocate term;
        store variable and order in the term;
      else
        compare the order of the variable in the token
        with the order in the variable in the term and
        update term if necessary;
    }
  }
return the list of term;
}
```

〈그림 10〉 알고리즘 *make-term*

```
make-item(list of handler and list of term)
{
  for each handler {
      allocate item;
      for each token connected to the handler {
          if the variable and order • the list of term
            add the variable and order to item;
    }
  }
return the list of item;
}
```

〈그림 11〉 알고리즘 *make-item*

3.2.2 변환 알고리즘

위 절에서 얻은 정보를 이용해서 연립 상미분방정식을 변환하는 과정은 각각의 미분방정식에서 적절한 최고 미분차항을 선택해서 그 항에 대해서 명시형으로 변환하는 것이다. 따라서 이 과정에서 가장 중요한 것은 최고 미분차항을 선택하는 것이다. 한 미분방정식에 최고 미분차항이 하나만 있으면 가장 쉬운 경우로 그 항에 대해서 명시형으로 변환하면 된다. 그러나 한 미분방정식에 두 개 이상의 최고 미분차항이 있으면, 어떤 항을 선택할 것인지를 결정해야하는 어려움이 있다. 따라서 현 시점의 PCSL에서는 명시형으로 변환하는 연립 상미분방정식의 형태를 제한하였다. PCSL에서는 다음과 같은 우선순위를 가지고 최고 미분차항을 선택해서 명시형으로 변환한다.

1. 미분방정식에 하나이상의 미분차항이 있는 경우, 그 식을 명시형으로 바꾼다.
2. 미분방정식에 하나이상의 종속변수에 대한 최고 미분차항이 있는 경우라도, 다른 식에서 변환되지 않은 최고 미분차항이 하나라면 그 항에 대해서 명시형으로 변환한다.
3. 어떤 최고 미분차항에 대해서 미분방정식이 명시형으로 변환되었으면, 모든 미분방정식에 대해서 그 최고 미분차항이 변환되었음을 알린다.
4. 이러한 과정을 반복해서 모든 미분방정식이 명시형으로 변환되도록 한다.

위의 과정을 구현한 알고리즘 *transform*은 그림 12에 나타나 있다.

```
transform(list of handler, list of term, list of item)
{
  while all ODE are not transformed {
    for each item {
        if item → cnt=1
          transform the ODE to explicit form
           according to item → var;
          store the result to term;
          for all item {
            remove the variable from item;
            reduce item → cnt, if necessary;
          }
    }
  }
  return the list of term;
}
```

〈그림 12〉 알고리즘 *transform*

이와 같은 과정을 거쳐서 변환된 명시형의 미분방정식

들은 위 장에서 제시된 알고리즘을 이용해서 각각의 차수 개만큼의 미분방정식들로 변환된다.

3.3 f.c의 생성

f.c는 변수 y_1를 입력으로 받아서 Integ(y_1)의 값을 출력하는 부프로그램이다. 예를 들어 모델에 대한 미분방정식으로 $f(x, y^{(1)}, \cdots, y^{(N)})$이 입력되었을 경우, 변환기는 f를 $y^{(N)}$에 대해서 명시형으로 변환하고, 그 변환된 식에서 각 $y^{(i)}$를 y_i로 치환해서 $f'(x, y_1, \cdots, y_{N-1})$라는 식을 생성한다. f.c는 $0 \leq i \leq N-1$인 y_i에 대해서 y_{i+1}의 값을 출력하고, y_{N-1}에 대해서는 Integ(f'의 값을 출력해야 한다. 이러한 기능이 요구되는 f.c에는 그림 13과 같은 알고리즘을 갖는 함수가 생성되어야 한다.

```
function(y_i)
{
  if(0≤i<N-1)
    return(y_{i+1}
  if(i = N-1)
    return(f'(x,y_1,⋯,y_{N-1}))
}
```

〈그림 13〉 함수 y_i

위의 *function*에서 미분방정식이 바뀜에 따라 변해야 하는 부분은 N과 f'이다. 따라서, *f.c*를 생성하는 것은 알고리즘 *function*을 구현한 C 프로그램에서 미분방정식이 바뀔 때마다 N과 f'에 대한 값만 바꾸어서 생성하면 된다.

4. 생성기의 구현

생성기에서는 digital-analog simulation 기법을 이용하는 모의 실험 알고리즘을 구현한 C 프로그램인 **main.c**를 생성한다. PCSL 처리 시스템에서는 미분방정식의 해를 알고리즘으로 Hamming의 다단계 방법을 이용한다[5]. Hamming의 방법에서는 다음과 같은 예측자, 보정자 공식을 이용해서 원하는 답을 얻는다.

$$y_{i+1}^{p} = y_{i-3} + \frac{4h}{3}(2f_i - f_{i-1} + 2f_{i-2})$$

$$y_{i+1}^{m} = y_{i+1}^{p} - \frac{112}{121}(y_i^{p} - y_i^{c})$$

$$y_{i+1}^{c} = \frac{1}{8}[9y_i - y_{i-2} + 3h(f_{i+1}^{m} + 2f_i - f_{i-1})]$$

$$y_{i+1} = y_{i+1}^{c} + \frac{9}{121}(y_{i+1}^{p} - y_{i+1}^{c})$$

위의 식에서 p는 예측자(predictor)를, m은 조정자(modifier)를 c는 보정자(corrector)를 의미한다. 입력된 미분방정식에 대해서 위의 식을 적용해서 만들어진 C 프로그램의 알고리즘은 그림 14와 같다.

```
Hamming
{
  Store initial value of X to X
  i=0
  Calculate f_i f_{i+1} f_{i+2} using Runge-Kutta method
  while(X < final value of X) {
      Calculate y_{i+1}^p
      Calculate y_{i+1}^m
      Calculate y_{i+1}^c
      Calculate y_{i+1}
      Calculate f_{i-1}
      Increase i
      Increase X by step size
  }
}
```

〈그림 14〉 알고리즘 f.c

생성기는 입력되는 미분방정식의 종류와 형태에 따라서 위의 알고리즘을 구현한 C 프로그램인 **main.c**를 자동으로 생성하기 위하여 다음과 같은 일을 한다.

1. 입력된 미분방정식의 수가 하나인 경우, 종속변수의 최고미분차수만큼의 새로운 변수가 생성되므로, 그 각각에 대해서 y, y^p, y^m, y^c, f등 필요한 자료 구조를 선언한다.
2. 입력된 미분방정식이 여러 개인 경우(연립인 경우), 먼저 입력된 미분방정식에서 종속변수의 수를 파악하고, 그 각각의 변수에 대해서 위의 과정을 수행한다.
3. 새로 생성된 각각의 변수에 대해서 위의 알고리즘이 적용되도록 한다.

따라서 입력되는 미분방정식이 변함에 따라서 **main.c**에서 일어나야 하는 변화는 그 미분방정식의 수와 종속

변수의 최고 미분차수에 의해서 결정된다.

5. 실행기의 구현

실행기는 번역기와 생성기에서 만든 f.c와 main.c를 컴파일해서 실행 파일을 만들고, 그것을 실행해서 결과를 얻는 역할을 한다. 먼저, f.c와 main.c를 컴파일하기 위해서 실행기에서는 입력된 PCSL 프로그램에서 모의 실험 수행에 필요한 여러 가지의 정보들을 추출해서 main.h라는 헤더 파일을 만든다. 따라서 모의 실험을 수행하기 위해서는 main.c와 f.c, main.h를 컴파일해서 실행해야 한다. 그 결과는 PCSL 프로그램에서 미리 선언한 OUTPUT 변수의 값이 수치의 형태로 출력 파일에 저장되도록 한다.

6. 구현결과

6.1 하나의 상미분방정식의 예

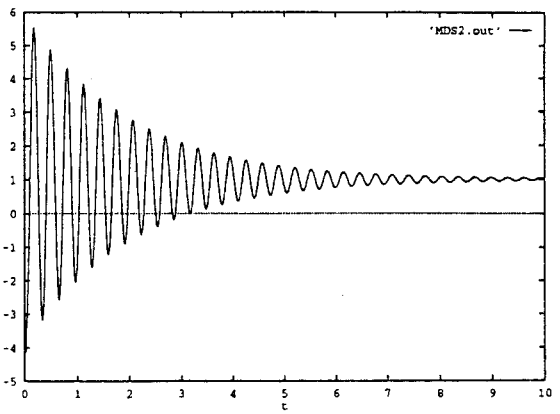
6.1.1 선형 상미분방정식의 예

하나의 선형 상미분방정식의 예로는 mass-damper-spring system을 들었다[9]. 여기에서 쓰이는 미분방정식은 $M^*x''+D^*x'+K^*x-F=0$이다. 초기조건으로는 $y(0)=0.0$, $y'(0)=0.0$을 주었다. 이 미분방정식을 풀기위한 PCSL 프로그램은 그림 15에서 주어진다.

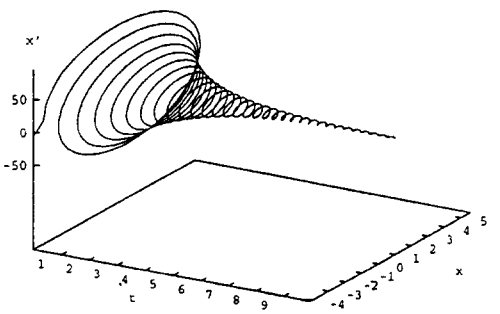
```
BEGIN
VARIABLE
      INDEP : t
      DEP : t
      SUB :
CONSTANT
      M : 1.0
      D : 1.0
      K : 400
      F : 400
RANGE
      R_BEGIN : 0
      R_END : 10
      R_STEP : 0.0001
INITIALS
      x_INIT_0 : 0.0
      x_INIT_1 : 0.0
MODEL
      "M*x"+D*x'+K*x-F=0"
SUBMODEL
END
```

〈그림 15〉 MDS system을 풀기 위한 PCSL 프로그램

실행결과로 얻은 t와 x의 2차원 그래프와 t,x,x'의 3차원 그래프는 그림 16과 그림 17에 나타나있다.



〈그림 16〉 Mass-Damper-Spring system의 2차원 그래프



〈그림 17〉 Mass-Damper-Spring system의 3차원 그래프

6.1.2 비선형 상미분방정식의 예

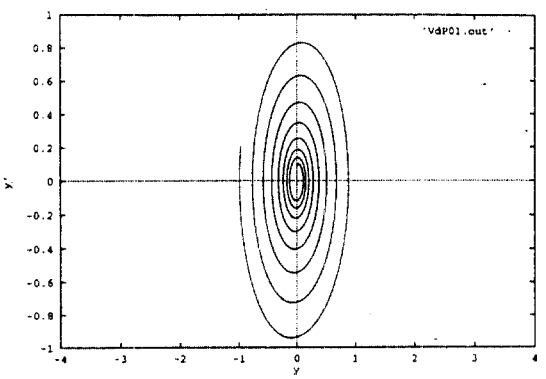
하나의 비선형 상미분방정식의 예로는 Van der Pol 방정식을 들었다. 먼저 Van der Pol 방정식 $y''-u^*(1-y^*y)^*y'+y=$

0을 푸는 PCSL 프로그램은 그림 18과 같다[7].

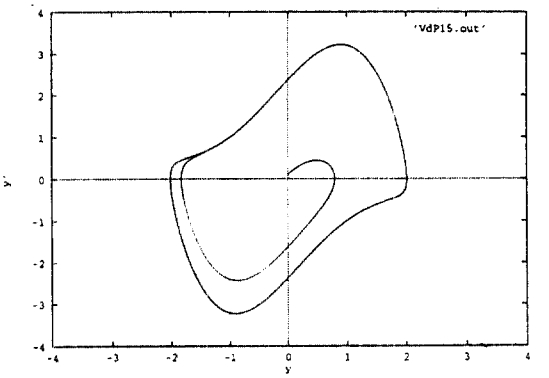
```
BEGIN
VARIABLE
    INDEP : t
    DEP : y
    SUB :
CONSTANT
    u : 0.1
RANGE
    R_BEGIN : 0
    R_END : 10
    R_STEP : 0.0001
INITIALS
    y_INIT_0 : 0.0
    y_INIT_1 : 0.1
MODEL
    "y"-u*(1-y*y)*y'+y=0"
SUBMODEL
END
```

〈그림 18〉 Van der Pol 방정식을 풀기 위한 PCSL 프로그램

이 프로그램에서 u의 값을 0.1과 1.5, 두 가지로 놓고 수행하여 결과를 얻었다. 각각의 결과에 대한 그래프는 그림 19과 그림 20에 나타나있다.



〈그림 19〉 u=0.1일 때 Van der Pol 방정식의 그래프



〈그림 20〉 u=1.5일 때 Van der Pol 방정식의 그래프

6.2 연립 상미분방정식의 예

연립 상미분방정식의 예로는 해가 주어진 두개의 미분방정식으로 된 문제와 mixing tank 문제를 들었다[3].

먼저 해가 주어진 두개의 방정식 문제는 다음과 같다.

$$y''_1-2*y'_2-y_1=0$$
$$y''_1-y'_1+x*y_2=0$$

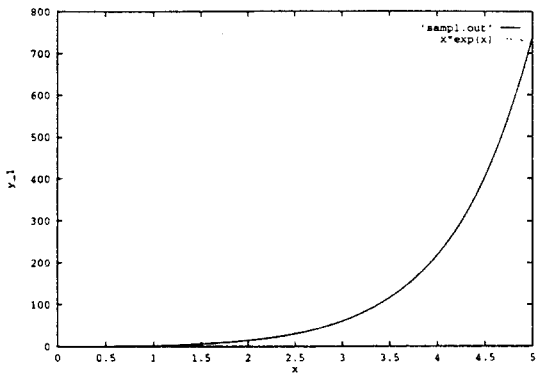
이 경우에 y_1은 $x*e^x$의 해를 갖고, y_2는 e^x의 해를 갖는다. 초기 조건으로는 $y_1(0)=0.0$, $y'_1(0)=1.0$, $y_2(0)=1.0$, $y'_2(0)=1.0$이다. 위의 두 방정식을 푸는 PCSL 프로그램은 그림 21과 같다.

```
BEGIN
VARIABLE
    INDEP : x
    DEP : y_1, y_2
    SUB :
CONSTANT
RANGE
    R_BEGIN : 0.0
    R_END : 5.0
    R_STEP : 0.0001
INITIALS
    y_1_INIT_0 : 0.0
    y_1_INIT_1 : 1.0
    y_2_INIT_0 : 1.0
    y_2_INIT_1 : 1.0
MODEL
```

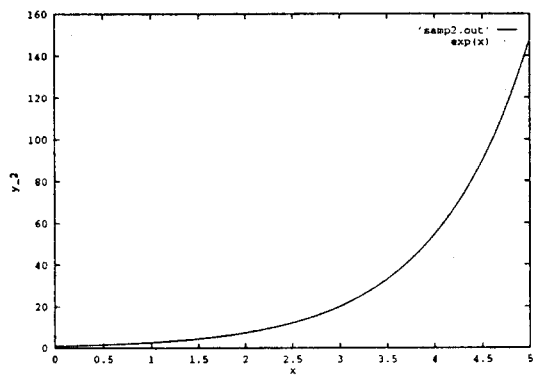
```
    "y_1" -2*y_2'-y_1=0"
    "y_2" -y_1'+x*y_2=0"
SUBMODEL
END
```

〈그림 21〉 연립 방정식을 풀기 위한 PCSL 프로그램

위 프로그램의 수행결과를 각각의 변수에 대해서 주어진 해와 비교한 그래프는 그림 22와 그림 23에 나타나있다.



〈그림 22〉 y_1과 $x*e^x$의 그래프



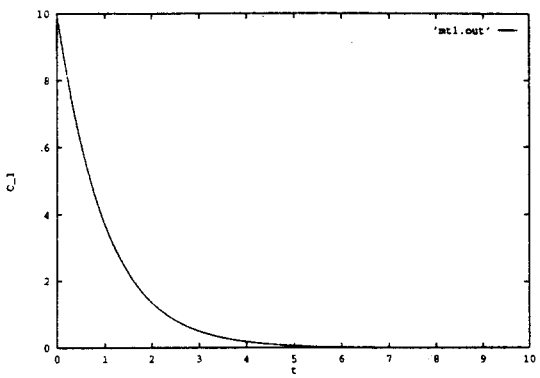
〈그림 23〉 y_2과 e^x의 그래프

그리고 mixing tank problem을 푸는 PCSL 프로그램은 그림 24와 같다.

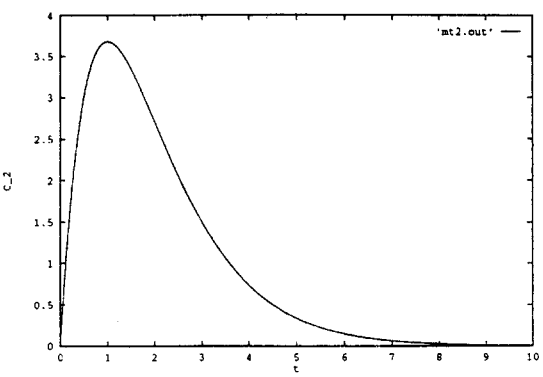
```
# This is a syntax of PCSL program

BEGIN
VARIABLE
    INDEP : t
    DEP : C_1, C_2
    SUB :
CONSTANT
RANGE
    R_BEGIN : 0.0
    R_END : 10.0
    R_STEP : 0.0001
INITIALS
    C_1_INIT_0 : 10.0
    C_2_INIT_1 : 0.0
MODEL
    "V*C_1'-L*C_0+L*C_1=0"
    "V*C_2'-L*C_1+L*C_2=0"
SUBMODEL
END
```

〈그림 24〉 mixing tank problem을 풀기 위한 PCSL 프로그램

계산 결과의 그래프를 그림 25과 그림 26에 나타나 있다.



〈그림 25〉 C—1의 그래프



〈그림 26〉 C—3의 그래프

7. 결론 및 앞으로의 개선 사항

본 논문에서 서술한 PCSL은 미분방정식으로 모델링되는 연속 체계를 모의실험 하는데에 있어서 사용자에게 편의성을 제공하기 위하여 개발되었다. PCSL에서 제공되는 방법은 사용자가 대상을 미분방정식으로 모델링한 후에 미분방정식과 계산에 필요한 몇몇 데이타를 언어의 형식으로 입력하면 자동으로 모의 실험을 행한 후, 그 결과를 사용자에게 제시하는 형식을 택하였다. 여기서 결과는 수치의 형태로 파일에 저장될 수도 있고, gunplot등의 도구를 이용해서 그래프로 그려질 수도 있다. 이 논문에서는 PCSL의 실행 예로 여러 가지 경우의 미분방정식들로 모델링되는 연속 체계에 대한 수행 결과를 그래프로 보였다.

먼저 처리할 수 있는 미분방정식의 범위를 편미분방정식까지 넓혀야 한다. 편미분방정식의 경우, 간단한 예로 Laplace 방정식, Poisson 방정식, 열전도 방정식, 파동 방정식등을 고려하고 좀 더 일반적인 경우에 대해서는 FDM이나, FEM등의 방법까지도 고려해야 한다.

그리고 사용자 인터페이스를 구현해야 한다. 현재는 shell 상에서 편집기를 이용해서 PCSL 프로그램을 입력하고 있으나, 사용자에게 더 큰 편의성을 제공하기 위해서는 PCSL의 문법을 가지고 그 형식을 사용자에게 제시하여 그 순서에 따라서 프로그램을 입력하면 되는 그래픽 사용자 인터페이스(GUI)를 제공하고, 또 미분방정식을 손쉽게 입력할 수 있도록 하는 도구(tool)도 제공해야 한다. 즉, PCSL 프로그램의 필요한 부분(예를 들면, CONST, INIT, OUTPUT)등을 윈도우 상에서 보여주고, 그 각각에 해당하는 변수와 값을 입력받도록 하는 GUI를 제공해야 한다. 그리고 모의 실험 결과를 사용자가 쉽게 이해할 수 있게 하는 도구역시 제공해야 한다.

참고문헌

[1] Maurice F.Aburdene, "Computer Simulation of Dynamic Systems", Wm.C.Brown Publishers, 1988.

[2] Chan Mo Park, Chang Kwan Yim, and Myung Soo Kim, "A Study on Development of User-friendly Continuous System Simulation Language and Application Software", *Proc. of 2nd Beijing International Conference on System Simulation and Scientific Computing*, Vol. Ⅱ, pp. 420-424, Beijing, 1992.

[3] Samuel D. Conte and Carl de Boor, "Elementary Numerical Analysis", McGraw Hill, 1980.

[4] Lee W. Johnson and R.Dean Riess, "Numerical Analysis", Addison Wesley, 1982.

[5] Edward E. L. Mitchell and Joseph S. Gauthier, "ACSL : Advanced Continuous Simulation Language - User Guide and Reference Manual", Mitchell and Gauthier Assoc., Concord, Mass, 1986.

[6] 박재년, "수치해석", 정익사, 1986.

[7] Erwin Kreyszig, "Advanced Engineering Mathematics, fifth edition", John Wiley & Sons, 1982.

[8] John V. Wait and DeFrance Clarke Ⅲ, "DARE-P User's Manual, Version 4.1", Dept. of Electrical & Computer Engineering, University of Arizona, Tucson, Ariz, 1976

[9] 임창관, "PCSL : 사용자 편의성을 고려한 연속체계 모의 실험 언어의 개발에 관한 연구", 포항공대 석사 학위 논문, 1993.

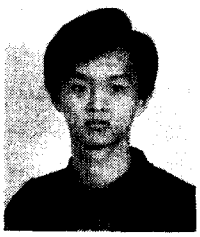
[10] 윤상엽, "전로조업의 동적모사", 포항공대 화공과 석사학위 논문, 1991.

[11] Francois E. Cellier, "Continuous System Modeling," Springer-verlag, 1990.

저자소개

박찬모

1958 서울대학교 화학공학과
1964 University of Maryland 화공학 석사
1969 University of Maryland 화공학 박사
1964~1969 University of Maryland 전산소 연구원
1969~1972 University of Maryland 전산학과 조교수
1973~1976 한국과학기술원 전산학과 부교수
1976~1979 National Biomedical Research Foundation 선임 연구원
1979~1989 Catholic University of America 전산과 교수 및 학과 주임
1990~현재 포항공대 전산학과 교수 및 학과 주임
1991~현재 포항공대 정보산업대학원장
1991~1992 한국시뮬레이션학회 초대회장
1993~현재 한국정보과학회장
관심분야: Image Processing, Computer Graphics, Computer Vision, System Simulation



임창관

1991 한국과학기술대학 전산학과 학사
1993 포항공과대학 전산학과 석사
1993~현재 삼성전자
관심분야: Computer Graphics, System Simulation

민경하

1992 한국과학기술대학 전산학과 학사
1992~현재 포항공과대학 전산학과 대학원 재학중
관심분야: System Simulation, Virtual Reality