Základy *gPROMS*

Vydání 2000

Primer for *gPROMS*

Heiko Briesen and Georg Schopfer
Lehrstuhl für Prozesstechnik der RWTH Aachen
Turmstraße 46, 52056 Aachen
Český překlad z angličtiny: Egon Eckert, ÚCHI, VŠCHT Praha, listopad 2000.

Obsah

1 Úvod

1.1 Proč používáme simulaci procesu

Se zvýšením výkonnosti počítačů v posledních desetiletích zaznamenává procesní simulace rostoucí oblibu. Dnes je to nepostradatelný pracovní nástroj k získávání znalostí o procesech ve všech fázích jejich života. Obecně musíme rozlišovat mezi dynamickou a stacionární simulací procesu. Stacionární simulace se používá při řešení problémů týkajících se počátečního stadia vývoje procesu a následného podrobného inženýrského zpracování jednotlivých jednotkových operací. Dovoluje také analýzu citlivosti procesu vzhledem k operačním parametrům případně jejich optimalizaci pro již fungující provoz.

Dynamická simulace procesu ale zahrnuje všechny tyto možnosti a ještě dále ohromně rozšiřuje oblast řešitelných problémů. Např. test stability zkoumaného procesu na změny vnějších parametrů stacionární simulací má malý význam, protože poskytne pouze jeden pracovní bod odpovídající dané sadě parametrů. V dynamické simulaci můžete specifikovat poruchy a testovat, zdali zkoumaný proces pracuje řádně nebo je nestabilní.

Pokud je třeba, aby zařízení pracovalo v nestabilním bodě, je třeba zvážit regulační strategie k udržení zařízení v požadovaných podmínkách. Jestliže chcete otestovat práci použitého regulátoru musíte rovněž použít dynamické simulace procesu, aby bylo vidět chovaní regulovaného systému. Rovněž regulační strategie, které vycházejí z modelu zařízení, se opírají o dynamickou simulaci procesu.

Je zřejmé, že vývoj a návrh vsádkových procesů vycházející z jejich modelu není možný na základě stacionární simulace, protože tyto procesy vykazují časově závislé chování už ze své definice. Totéž platí i pro chování kontinuálně provozovaných zařízení při najíždění a odstávce, které se často zkoumá z důvodu bezpečnosti.

Posledním příkladem, který by zde měl být zmíněn, je použití dynamické simulace k tréninku operátorů. Výuka provozního řídícího personálu se dnes obvykle provádí na tréninkových simulátorech provozu, jestliže je k dispozici jeho dostatečně dobrý model. Tak mohou operátoři získávat zkušenosti i v kritických situacích, aniž by hrozilo riziko destrukce nákladného zařízení.

1.2 Co je *gPROMS*

gPROMS je nástroj k získání odpovědí na otázky popsané v předcházející sekci. Dovoluje uživateli flexibilně sestavit model podle určitých syntaktických pravidel, který *gPROMS* použije k vytvoření simulačního experimentu. Takže *gPROMS* obsahuje v zásadě druh programovacího jazyka vytvořeného speciálně pro potřeby procesní simulace a dále výkonné numerické zázemí, které dovoluje uživateli se soustředit na mode-

lování a ne na numerické metody. (Porozumění numerickým metodám používaným v *gPROMS* má ale často zásadní význam k tomu, aby vaše simulace správně fungovala.)

Modelování chemicko-inženýrských procesů často zahrnuje použití diferenciálních rovnic. Diferenciální rovnice pocházejí z bilančních rovnic pro hmotu, energii (a hybnost) a udávají časový vývoj diferenciálních stavů (např. obsah složky v zadaném bilančním objemu). Tyto diferenciální rovnice doplňují sady algebraických rovnic popisujících např. fázové rovnováhy nebo regresní funkce pro fyzikální vlastnosti. Dohromady tyto rovnice tvoří systém DAE (differential-algebraic-equations), pro který jsou v *gPROMS* implementovány numerické metody.

Na rozdíl od ostatních komerčně dostupných programů pro dynamickou simulaci je *gPROMS* rovněž schopen pracovat s distribuovanými systémy. Distribuované systémy jsou charakterizovány skutečností, že v nich chování systému závisí i na prostorové souřadnici. To znamená, že často používaný předpoklad ideálně promíchávaného systému, který vede k tzv. systémům se soustředěnými parametry, zde neplatí. Kromě toho diferenciální a algebraické rovnice příslušných modelů obvykle obsahují parciální derivace vzhledem k prostorové souřadnici.

Nejobecnějším příkladem v procesním inženýrství může být trubkový reaktor, jak je uveden v následujícím odstavci.

V *gPROMS* se tato třída modelů nejdříve podrobí diskretizaci prostorových proměnných parciálně diferenciálně algebraického systému (the partial-differential-algebraic (PDAE) system) a pak se řeší vzniklý DAE systém.

1.3 Příklady

K ilustraci použití *gPROMS* jsou v rámci této příručky používány dva příklady. V jednom je uvažována jednoduchá míchací nádoba, ve které se směšují dvě složky. Pro ilustraci přístupu k distribuovaným systémům je použit jednoduchý reaktor s pístovým tokem.

1.3.1 Míchací nádoba s přepadem

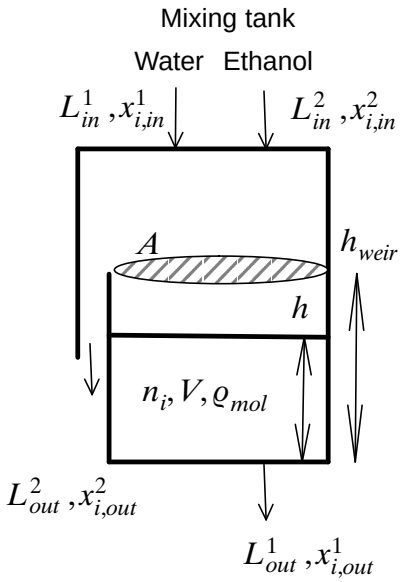
Na obr. 1 jsou znázorněny dva oddělené nástřiky vody a ethanolu, které se míchají v jednoduché nádobě. Jestliže hladina dosáhne výšky jezu, začne kapalina přetékat.

Model, který odráží chování nádoby (bez regulace hladiny) je dán následující soustavou rovnic (DAE):

$$\frac{dn_i}{dt} = L_{in1}x_{i,in1} + L_{in2}x_{i,in2} - L_{out1}x_{i,out1} - L_{out2}x_{i,out2}\ ,\ i = 1,2 \tag{1}$$

$$n_g = n_1 + n_2 \tag{2}$$

$$n_i = x_i\, n_g\ ,\ i = 1,2 \tag{3}$$



Obrázek 1: Míchací nádoba

$$
\begin{aligned}
x_i &= x_{i,out1} = x_{i,out2}\ ,\ i = 1,2 && (4)\\
V &= h \cdot A && (5)\\
n_g &= V \cdot \varrho_{mol} && (6)\\
L_{out2} &= \begin{cases} \alpha \left(h - h_{weir}\right)^{0.5} & : \quad h > h_{weir} \\ 0 & : \quad \text{else} \end{cases} && (7)\\
\varrho_{mol} &= \varrho_{mol}\left(x_i\right), && (8)
\end{aligned}
$$

s počátečními podmínkami

$$
\begin{aligned}
n_g\left(t=0\right) &= n_{g,0} && (9)\\
x_1\left(t=0\right) &= x_{1,0}, && (10)
\end{aligned}
$$

zadanými vstupy

$$
\begin{aligned}
x_{i,in1} &= x_{i,in1}\left(t\right)\ ,\ i = 1,2 && (11)\\
L_{in1} &= L_{in1}\left(t\right) && (12)\\
x_{i,in2} &= x^{i,in2}\left(t\right)\ ,\ i = 1,2 && (13)\\
L_{in2} &= L_{in2}\left(t\right) && (14)
\end{aligned}
$$

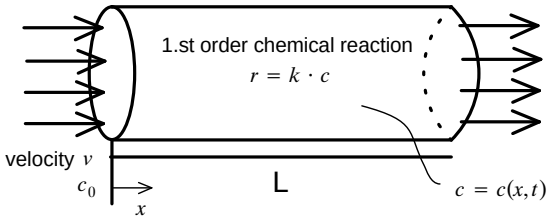
a zadanými parametry A, h_{weir} and α.

1.3.2 Reaktor s pístovým tokem

Jako příklad distribuovaného modelu je studován jednoduchý reaktor s pístovým tokem. Parciální diferenciální rovnice popisující reaktor s pístovým tokem je:

$$\frac{\partial c}{\partial x} + \frac{1}{v}\frac{\partial c}{\partial t} = -kc \tag{15}$$

Počáteční a okrajové podmínky jsou uvažovány ve tvaru:



Obrázek 2: Reaktor s pístovým tokem s reakcí prvého řádu.

$$c\,(x=0,t) \;=\; c_0\;;\; t \geq 0 \tag{16}$$
$$c\,(x,t=0) \;=\; c_0\;;\; 0 < x \leq L\,. \tag{17}$$

Jak je vidět z tohoto modelu, nezávisí koncentrace jen na čase, ale také na axiální souřadnici v reaktoru. V tomto příkladě jsme zvolili velmi jednoduchý problém popsaný jedinou parciální diferenciální rovnicí. Obecně ovšem mohou distribuované modely obsahovat rovněž obyčejné diferenciální a algebraické rovnice v soustavě s více než jedinou parciální diferenciální rovnicí.

2 Psaní vstupního souboru pro *gPROMS*

Simulační experiment se specifikuje v jediném vstupním souboru pro *gPROMS*. K vytvoření toho vstupního souboru můžete použít libovolného textového editoru. Vstupní soubor musí být uložen s extenzí `.gPROMS` (např. `tank.gPROMS`) v podadresáři `gPROMS/input`. Poté ho můžete vyvolat spuštěním tohoto vstupního souboru v *gPROMS*. Jak spustit *gPROMS* je popsáno v kapitole 3.1.

Struktura vstupního souboru k příkladu míchací nádoby je ukázána níže. Úplný vstupní soubor je uveden v dodatku A.1.

```
{
  Simple Mixing tank model (only material balance) with overflow
  weir.
  ...
```

```
}
#-------------------------------------------------------
# Declarations
#-------------------------------------------------------
  DECLARE
   TYPE
    ...
  END #DECLARE section
#-------------------------------------------------------
# Model description
#-------------------------------------------------------
  MODEL MixingTank
   PARAMETER
    ...
   VARIABLE
    ...
   EQUATION
    ...
  END # Model Tank
#-------------------------------------------------------
# Process description
#-------------------------------------------------------
  PROCESS mix1
   UNIT
    ...
   SET # Parameter values
    ...
   ASSIGN # Degrees of freedom
     ...
   INITIAL # Initial conditions
    ...
   SOLUTIONPARAMETERS
     ...
   SCHEDULE # Operating procedure
     ...
  END # PROCESS mix1
```

Vstupní soubor se skládá ze sekce `DECLARE`, popisu `MODEL` a popisu `PROCESS`. Je důležité poznamenat, že pořadí sekcí musí být voleno tak, aby všechno bylo deklarováno dříve, než je to použito. V sekci `DECLARE` se deklarují typy proměnných, sekce `MODEL` obsahuje rovnice a proměnné popisující fyzikální chování procesu a sekce `PROCESS` definuje za jakých podmínek zkoumaný proces pracuje.

Z příkladu jsou dále patrné komentáře objasňující obsah vstupního souboru. *gPROMS*

připouští dva typy komentářů. Komentáře zahrnující více řádek jsou v závorkách "{ }". Tyto komentáře mohou být vkládány do sebe. Jednořádkové komentáře začínají "#".

Popis jednoduchého dynamického systému, jako je míchací nádoba, se dá udělat ve vstupním souboru s jedinou sekcí `MODEL`, model je popsán pomocí soustavy rovnic DAE. Samozřejmě mohou být popsány rovněž složitější systémy spojením několika `MODEL`ů.

V dalším nejdříve objasníme, jak vypadají sekce základního vstupního souboru (sekce 2.1 - 2.3). Poté popíšeme jak definovat složitější modely pomocí spojování modelů (odstavce 2.4). Nakonec objasníme zacházení se systémy s distribuovanými parametry (odst. 2.5).

2.1 DECLARE

Vstupní soubor by měl vždy začínat komentářem vysvětlujícím, co soubor obsahuje. Rovněž by zde měla být informace o autorovi a historie úprav v případě složitějších modelů. Po tomto komentáři každý vstupní soubor pro *gPROMS* začíná sekcí `DECLARE`, v níž jsou deklarovány všechny typy proměnných použitých v následujících `MODEL`ech. To odpovídá pravidlu, že každý typ proměnné musí být deklarován předtím, než může být použit v `MODEL`u. Sekce `DECLARE` pro `tank.gPROMS` následuje:

```
DECLARE
  TYPE
    Mole          =   55  :  -1E5  :  1E5  UNIT = "kmole"
    MoleFlowrate  =  1.0  :  -1E5  :  1E5  UNIT = "kmole/min"
    Length        =  1.0  :     0  :   10  UNIT = "m"
    Volume        =  1.0  :   0.0  :  500  UNIT = "m3"
    MolarFraction =  0.5  :     0  :  1E0  UNIT = "-"
    MolarDensity  =   50  :     0  :  100  UNIT = "kmol/m3"
END #DECLARE section
```

Každá deklarace typu proměnné obsahuje následující informaci:

variable type name = *default value* : *lower bound* : *upper bound* *unit of measurement* (nepovinná)

Např. deklarace

```
MoleFlowrate = 1.0 : -1E5 : 1E5 UNIT = "kmole/min"
```

jednoduše udává, že proměnné typu `MoleFlowrate` mají implicitní hodnotu 1.0, ležící mezi -10^5 a 10^5 a jsou měřeny v *kmol/min*. Implicitní hodnota bude použita jako počáteční odhad pro jakýkoliv iterační výpočet, pokud není nahrazena v sekci `PROCESS PRESET` (viz 2.3.5). Konvergenční vlastnosti inicializačních výpočtů závisejí silně na počátečním odhadu.

Horní a dolní mez nutí hodnoty proměnných tohoto typu, aby ležely v zadaném intervalu. Jestliže hodnoty proměnné překročí tyto hranice, simulace se ukončí. To nabízí možnost vyzkoušet, zdali střední hodnoty proměnných jsou fyzikálně smysluplné. *Jednotky* jsou nepovinná specifikace, která se používá v *gPROMS* při výstupu hodnot příslušného typu.

Poznámka 2.1 *Poznamenejme, že jednotky vůbec neovlivňují výpočet. Rovněž není automaticky testována konsistence rovnic, jednotky se používají pouze při výstupu výsledků. Věnujte proto péči prověřování rovnic na konsistenci jednotek, protože ta je jednou z nejčastějších příčin simulačních chyb.*

2.2 Sekce MODEL

Sekce `MODEL` obsahuje popis fyzikálního chování daného systému vyjádřený pomocí proměnných a rovnic, které se k němu vztahují. Základní struktura sekce `MODEL` je:

`MODEL` *ModelName*

`PARAMETER`

...Parameter declarations...

`VARIABLE`

...Variable declarations...

`EQUATION`

...Equation declarations...

`END`

Jednoduchá sekce `MODEL` je uvedena v příkladu míchací nádoby. Poznamenejme, že čísla řádků nejsou součástí vstupního souboru *gPROMS* a jsou uvedena pouze pro účely odkazování se na ně.

```
1   MODEL MixingTank
2
3     PARAMETER
4       NoComp              AS INTEGER
5       Area                AS REAL
```

```
6       WeirHeight          AS REAL
7       Alpha               AS REAL
8
9     VARIABLE
10      FlowIn1, FlowIn2    AS MoleFlowRate
11      FlowOut1, FlowOut2  AS MoleFlowRate
12      HoldUp              AS ARRAY (nocomp) OF Mole
13      xIn1,xIn2           AS ARRAY (nocomp) OF MolarFraction
14      xOut1,xOut2,xTank   AS ARRAY (nocomp) OF MolarFraction
15      TotalHoldUp         AS Mole
16      RhoMolComp          AS ARRAY (nocomp) OF MolarDensity
17      RhoHelp             AS ARRAY (nocomp) OF MolarDensity
18      RhoMol              AS MolarDensity
19      Vol                 AS Volume
20      Level               AS Length
21
22    STREAM
23      Inlet1 : FlowIn1, xIn1      AS MAINSTREAM
24      Inlet2 : FlowIn2, xIn2      AS MAINSTREAM
25      Outlet1: FlowOut1,xOut1     AS MAINSTREAM
26      Outlet2: FlowOut2,xOut1     AS MAINSTREAM
27
28    EQUATION
29
30      # Mass balance
31      $HoldUp=FlowIn1*xIn1+FlowIn2*xIn2-FlowOut1*xOut1
32                                       -FlowOut2*xOut2;
33
34      # molar concentrations
35      # Example for the FOR construct
36      FOR i := 1 TO NoComp DO
37          Holdup(i) = TotalHoldUp * xTank(i) ;
38      END
39
40      TotalHoldUp = SIGMA(HoldUp);
41
42      # Level
43      Vol = Level * Area;
44      Vol * RhoMol = TotalHoldUp;
45
46      # well-mixedness in tank
47      xTank = xOut1;
48      xTank = xOut2;
49
```

```
50      # Overflow over weir
51      IF Level > WeirHeight THEN
52           FlowOut2 = Alpha * SQRT(Level - WeirHeight);
53      ELSE
54           FlowOut2 = 0;
55      END # IF
56
57      # Molar Density of mixture
58      RhoHelp = xTank / RhoMolComp;
59      RhoMol = 1 / SIGMA(RhoHelp);
60
61  END # Model Tank
```

2.2.1 PARAMETER

Parametry sekce `MODEL` jsou časově neměnné konstanty, které odpovídají hodnotám, které *nebudou* počítány v simulaci. Musejí být deklarovány v sekci `MODEL PARAMETER` a jejich hodnoty musejí být specifikovány v sekci `PROCESS SET`. V sekci `PARAMETER` `MODELu MixingTank`

```
PARAMETER
  NoComp              AS INTEGER
  Area                AS REAL
  WeirHeight          AS REAL
  Alpha               AS REAL
```

můžeme pozorovat, že každý parametr je dán svým jménem a deklarován buď jako `INTEGER` nebo `REAL`. Jméno se používá v sekci `MODEL` a `PROCESS` jako odvolávka na tento parametr. V *gPROMS* se používá následující konvence pro jména: Jména musejí začínat písmenem a mohou obsahovat písmena, číslice a podtržítka. Poznamenejme, že jazyk *gPROMS nerozlišuje* velká a malá písmena tj. *NoComp* a *nocomp* jsou uvažovány jako identické.

2.2.2 VARIABLE

Proměnné (variables) reprezentují veličiny, které popisují časově závislé chování systému. V příkladu *MixingTank* veličiny `Flowin1, Flowin2, Holdup`, a `Level` jsou proměnnými `MODELu`. V sekci `VARIABLE` jsou deklarovány v příslušných typech definovaných v `DECLARE`.

```
VARIABLE
    FlowIn1, FlowIn2    AS MoleFlowRate
```

```
HoldUp              AS ARRAY (NoComp) OF Mole
Level               AS Length
...
```

Zde je proměnná `HoldUp` definována jako pole s délkou určenou parametrem `NoComp`. Je možné používat i dvourozměrná pole (např. `StoechCoeff AS ARRAY (NoComp, NoReac) OF Coefficient`).

2.2.3 EQUATION

Diferenciální a algebraické rovnice, které určují průběh proměnných deklarovaných v sekci `VARIABLE` jsou zadány v sekci `EQUATION`.

Podívejme se na sekci `EQUATION` v příkladu míchací nádoby. Všimněme si nejdříve krátkých komentářů, které by měly upozornit na účel každé rovnice. Dále můžeme pozorovat několik rysů zadávacího jazyka *gPROMS*, který podporuje několik způsobů vyjádření vztahů mezi poli. Na řádkách 58 a 59 je použit mechanismus nazývaný automatická expanze. Rovnice na řádce 58 se týká polí `RhoHelp`, `xTank`, a `RhoMolComp` s délkou `NoComp`. Za předpokladu, že `NoComp` se rovná 3, *gPROMS* rozšíří automaticky uvedený výraz na

```
RhoHelp(1) = xTank(1) / RhoMolComp(1);
RhoHelp(2) = xTank(2) / RhoMolComp(2);
RhoHelp(3) = xTank(3) / RhoMolComp(3);
```

Expanze na řádce 59 používá zabudovanou funkci `SIGMA` k sečtení všech prvků pole `RhoHelp`. Ta může být použita i na části polí jako např. `RhoHelp(1:2)`, `xTank(1:2)`, které by byly expandovány pouze na první dvě nahoře uvedené rovnice.

Všude, kde by automatická expanze nevedla k požadovaným rovnicím, mohou být vztahy mezi prvky polí rovněž vyjádřeny ve formě konstrukce `FOR`, jako je tomu na řádcích 27-29. Proměnná počitadla konstrukce `FOR` by měla mít jedinečné jméno a může být použita pouze v rovnicích v této konstrukci. Kromě toho může být specifikován i přírůstek kroku. Např.

```
FOR i := 1 TO 5 STEP 2 DO
    ....
END
```

přiřadí počítadlu `i` hodnoty 1, 3 a 5. Standardní přírůstek je 1. Všimněme si, že na řádcích 36-38 konstrukce `FOR` není potřebná a může být rovněž napsána s využitím automatické expanze:

```
Holdup = TotalHoldUp * xOut;
```

Diferenciální rovnice na řádce 31 popisuje časovou změnu proměnné `HoldUp`, jak to odpovídá hmotnostní bilanci. V *gPROMS* se používá symbolu "`$`" pro diferenciální operátor $\partial/\partial t$.

Na řádcích 50 až 55 je specifikována závislost přetoku přes jez, takže výstupní průtok `FLowOut2` závisí lineárně na druhé odmocnině rozdílu mezi výškou hladiny a výškou jezu. Abychom se vyhnuli zápornému přetoku pro hladiny menší než je výška jezu, je použito konstrukce `IF-THEN-ELSE`. Ta specifikuje, že rovnice má být použita pouze tehdy, je-li hladina výše než je jez (`IF-THEN`), jinak se dosadí `FLowOut2` rovno 0 (`ELSE`).

V neposlední řadě nabízí *gPROMS* některé zabudované funkce jako `SQRT` na řádce 52. Některé důležité z nich jsou uvedeny v tabulce 1.

Tabulka 1: Zabudované funkce

Identifikátor	*gPROMS* Funkce
`SIN`	Sinus argumentu v radiánech
`EXP`	Exponent argumentu
`SQRT`	Druhá odmocnina z argumentu
`LOG`	Přirozený logaritmus argumentu
`SIGMA`	Sumace prvků pole

Poznámka 2.2 *Kompletní seznam zabudovaných (intrinsic) funkcí je uveden v gPROMS Introductory User Guide .*

2.2.4 Použití rozhraní pro fyzikální vlastnosti

gPROMS nemá zabudovanou vlastní databázi fyzikálních vlastností. Vedle možnosti uložení rovnic pro fyzikální vlastnost do vstupního souboru podobně, jako je to uděláno ve zde uváděných příkladech, je možno rovněž použít rozhraní ke standardním programům pro výpočty fyzikálních vlastností, jako jsou např. IK-CAPE nebo Multiflash.

2.3 PROCESS

Po sestavení modelů je třeba do vstupního souboru začlenit i sekci `PROCESS`, která obsahuje informaci o specifické formulaci problému. Obvykle je `MODEL` nedourčen, dokud nejsou zadány jeho některé parametry, počáteční podmínky a stupně volnosti. Zatímco model sám může popisovat fyzikální chování velké řady problémů, specifikace uvedená

v této sekci určuje aktuální *simulační experiment*. Procesní sekce obsahuje různé podsekce, které všechny budou popsány v následujícím odstavci. Syntaxe sekce `PROCESS` je následující:

```
PROCESS <Name>
  UNIT
   <setting up instance of the model(s)>
  MONITOR
   <specifying variables to be monitored>
  SET
   <setting of model parameters>
  ASSIGN
   <specifying the degrees of freedom>
  PRESET
   <setting starting values for the iterative solution>
  INITIAL
   <specifying initial conditions>
  SOLUTIONPARAMETERS
   <controlling the solution algorithms>
  SCHEDULE
   <specifying operation>
END
```

2.3.1 UNIT

Nejdříve je třeba určit zařízení v problému, které se bude počítat. V našem příkladu míchací nádoby se to provede takto:

```
UNIT
  Mixer1 AS MixingTank
```

Tímto příkazem jsme vytvořili případ modelu `MixingTank` s názvem `Mixer1`.

2.3.2 MONITOR

V sekci `MONITOR` je možno vybrat proměnné, jejichž hodnoty nás zajímají v konečném výsledku. Bez zadání sekce `MONITOR`, se ukládají všechny proměnné pro účely kreslení grafu a hodnocení řešení. Omezení na určitou sadu proměnných, které nás zvláště zajímají, je užitečné např. v případě, kdy jsou zkoumány velké distribuované problémy, kdy počet proměnných snadno dosáhne několika tisíc. Soubory, do kterých se ukládají výsledky, by pak byly velmi velké a práce s nimi by vyžadovala velkou paměť počítače

a narůstal by i výpočetní čas. Při odkazech na proměnné se používá tečkové notace obvykle používané v objektově orientovaných programovacích jazycích: *InstanceName.VariableName*.

Příklad:

```
MONITOR
     Mixer1.FlowOut1;
     Mixer1.FlowOut2;
     Mixer1.xTank;
     Mixer1.Level;
```

Poznámka 2.3 *Jestliže jsou požadovány výsledky simulace pouze v určitém časovém intervalu je možno monitorování proměnné zapnout a vypnout rámci sekce* `SCHEDULE` *(viz gPROMS Introductory User Guide).*

2.3.3 SET

Sekce `SET` obsahuje specifikace parametrů použitého zařízení. Zadáním proměnných v podsekci `SET` sekce `PROCESS` je možno přepsat standardní hodnoty zadané v sekci `MODEL`. Pro dosazení parametrů se používá přiřazovacího operátoru ':=':

```
SET # Parameter values
  Mixer1.NoComp := 2;
  Mixer1.Area := 1;
    .....
```

2.3.4 ASSIGN

Obvykle má model méně rovnic než proměnných. Rozdíl v počtu proměnných a nezávislých rovnic představuje počet stupňů volnosti modelu. Bez zadání těchto stupňů volnosti není problém plně určen a zůstává neřešitelný.

Specifikace stupňů volnosti ve vstupním souboru pro *gPROMS* se dělá v sekci `ASSIGN`. Např.

```
ASSIGN # Degrees of freedom
  WITHIN Mixer1 DO
      FlowIn1 :=10;
      xIn1(1) :=1;
      xIn1(2) :=0;
      ....
  END # WITHIN Mixer1
```

dosadí stupně volnosti pro Mixer. Konstrukce `WITHIN ... END` dovoluje vyhnout se *InstanceName* před jménem proměnné.

Tato přiřazení mohou být změněna v sekci `SCHEDULE` např. pro účely zkoumání chování modelu na změnu v hodnotách některých proměnných.

2.3.5 PRESET

V případě velkých vysoce nelineárních problémů může konvergence nelineárního solveru představovat vážný problém. Jak již bylo zmíněno v sekci 2.1 může být tento problém zmírněn zadáním rozumných mezí a počátečními odhady proměnných v sekci `DECLARE`. Touto cestou je možno ovšem zadat pouze jedinou hodnotu jako počáteční odhad pro všechny proměnné stejného typu.

Sekce `PRESET` dovoluje přiřadit odhady pro každou proměnnou zvlášť. Např.:

```
PRESET
  Mixer1.Level := 1;
  ...
```

Pečlivost a ohled na fyzikální realitu dovolují nalézt dobré počáteční hodnoty, ze kterých výpočet konverguje. Nicméně vyžaduje to inženýrské myšlení a zkušenosti a zdaleka to není v simulaci jednoduchý úkol.

Pro mnoho proměnných může být obtížné odhadnout dobré počáteční hodnoty. V tomto případě je obvyklou praxí dělat inicializační výpočty pomocí zjednodušené verze modelu. Výsledky těchto výpočtů je možno použít pak jako odhady pro inicializaci složitějšího modelu. V tomto případě budete mít dva `PROCESS`y. V prvním se provedou inicializační výpočty se zjednodušeným modelem a výsledky se uloží do datového souboru. To se udělá v sekci `PROCESS SCHEDULE` (viz sekce 2.3.8). Ve druhém `PROCESS`u je možno tento datový soubor použít v sekci `PRESET`. To se provede příkazem `RESTORE`:

```
PRESET
  RESTORE "filename";
```

V mnoha případech může být účelné uložit rovněž výsledky složitého `MODEL`u. Pokud poté změníte `MODEL`, mohou být tyto hodnoty opět použity jako počáteční hodnoty pro inicializaci modifikovaného `MODEL`u.

2.3.6 INITIAL

Jak víte z matematiky, je každá diferenciální rovnice numericky řešitelná pouze tehdy, jsou-li specifikovány počáteční podmínky. Ty určují stav systému na začátku zkoumaného časového intervalu. To se řeší v *gPROMS* v rámci sekce `INITIAL`. Protože *gPROMS*

je v prvé řadě určen pro řešení diferenciálně algebraických systémů (DAE), je zde větší volnost ve volbě počátečních podmínek než při řešení systémů obyčejných diferenciálních rovnic (ODE). Pro ODE musí být každé diferenciální proměnné přiřazena určitá hodnota, zatímco pro DAE může být být počáteční podmínka formulována jako dodatečná rovnice, která musí být splněna v čase t=0. Samozřejmě tato dodatečná rovnice musí doplňovat systém rovnic tak, aby byl řešitelný v čase t=0.

Příklad pro použití diferenciálních proměnných k inicializaci:

```
INITIAL
  HoldUp(1) = 55.39;
  HoldUp(2) = 0;
```

Příklad obecnější inicializace

```
INITIAL # Initial conditions
  WITHIN Mixer1 DO
       Level = 1;
       TotalHoldUp = 55.39;
  END # WITHIN Mixer1
```

Existuje také možnost inicializovat výpočet ze stacionárního stavu systému použitím:

```
INITIAL
  STEADY_STATE
```

2.3.7 SOLUTIONPARAMETERS

Existuje řada parametrů, které je možno použít k ovlivnění chování *gPROMS* při hledání řešení. Zde jsou uvedeny pouze některé nejužitečnější:

```
SOLUTIONPARAMETERS
  BlockDecomposition   := ON ;
  Integrator           := "SRADAU" ;
  LASolver             := "MA28" ;
  OutputLevel          := -1 ;
  ReportingInterval    := 2;
```

BlockDecomposition Použití `BlockDecomposition` k řešení vzniklého nelineárního systému rovnic. `BlockDecomposition` může být dosazeno jako `INITIAL`, `ON` a `OFF`, což znamená použití blokové dekompozice pouze pro inicializaci, stále nebo nikdy.

Integrator Jsou k dispozici dva integrátory `"SRADAU"` a `"DASOLV"`. Přednosti jednoho algoritmu vůči druhému závisejí silně na typu problému, takže nelze dát žádné obecné doporučení.

LASolver Jsou k dispozici dvě metody k řešení vzniklého systému lineárních rovnic. Je možno specifikovat buď `"MA28"` nebo `"MA48"`.

OutputLevel Pro výstup výsledků je možno specifikovat úroveň podrobnosti. Jsou povoleny celočíselné hodnoty v rozmezí od -1 (minimální výstup) do 6 (maximální výstup) Vyšší úroveň může být užitečná zejména při analýze chyb, zatímco při standardní simulaci je generovaný výstup na vyšší úrovni podrobnosti nepřijatelně dlouhý a složitý.

ReportingInterval Tato hodnota specifikuje časový interval, pro který je řešení ukládáno do souboru.

Poznámka 2.4 *Úplný seznam parametrů lze nalézt v gPROMS Introductory User Guide .*

2.3.8 SCHEDULE

V sekci `SCHEDULE` se specifikuje provozní strategie. Ta zahrnuje dobu simulace a změny vstupních proměnných. Sekce se skládá z jedné nebo více úloh (tasks). Řada elementárních úloh je uživateli standardně k dispozici, zatímco složitější úlohy mohou být specifikovány uživatelem ve zvláštní sekci. Nejdůležitější standardní elementární úlohy jsou ukázány v následujícím příkladě:

```
SCHEDULE # Operating procedure
  SEQUENCE
      CONTINUE FOR 20
      RESET
         Mixer1.FlowIn1 := 12;
      END
      CONTINUE FOR 20
      .....
```

Uložení hodnot proměnné do souboru v zadaném okamžiku uvnitř `SCHEDULE` se děje příkazem `SAVE`. Např.:

```
SCHEDULE # Operating procedure
    SEQUENCE
        CONTINUE FOR 0
        SAVE "filename"
    END  #sequence
```

Se syntaxí

SAVE <*vartype*> "filename"

můžete také uložit pouze podmnožinu proměnných podle specifikace <*vartype*> jako `state`, `algebraic` nebo `input` (nebo jako kombinace těchto oddělených ",").

Poznámka 2.5 *Všechny elementární úlohy a definici a použití uživatelem definovaných úloh lze nalézt v gPROMS Introductory User Guide .*

2.4 Složené modely

Spíše než definování velkých modelů v jediné sekci `MODEL`, *gPROMS* podporuje spojování několika modelů do jednoho celkového modelu. To zajišťuje, že velikost sekce `MODEL` zůstává taková, že sekce je přehledná a jednotlivé `MODEL`y mohou být opětně používány.

K tomu účelu *gPROMS* dovoluje deklaraci `MODEL`ů vyšší úrovně, které obsahují případy dříve deklarovaných `MODEL`ů. `MODEL`y vyšší úrovně obsahují, nehledě na své vlastní proměnné, parametry a rovnice, rovněž proměnné, parametry a rovnice `MODEL`ů, které jsou jejich součástí. Toto je podrobněji popsáno v odstavci 2.4.1.

gPROMS umožňuje navíc mechanismus nazvaný `STREAM`, který dovoluje pohodlné spojení modelů. Použití `STREAM`ů bude vysvětleno v sekci 2.4.2.

2.4.1 Deklarace modelů vyšší úrovně

Předpokládejme, že chceme modelovat provoz skládající se z řady tří míchacích nádob za sebou. V tomto případě chceme vytvořit `MODEL` vyšší úrovně, který specifikuje celý systém. Nechť se tento `MODEL` jmenuje `MixingTankCascade`.

```
MODEL MixingTankCascade

  UNIT
    Tank1 AS MixingTank
    Tank2 AS MixingTank
    Tank3 AS MixingTank

  EQUATION
    Tank2.FlowIn2 = Tank1.FlowOut1;
    Tank3.FlowIn2 = Tank2.FlowOut1;
    Tank2.xIn2 = Tank1.xOut1;
    Tank3.xIn2 = Tank2.xOut1;

END  #MODEL MixingTankCascade
```

V sekci UNIT je specifikováno, jaké technické zařízení se používá k sestavení celého provozu. V uvedeném příkladě se provoz skládá ze tří míchacích nádob s názvy Tank1, Tank2 a Tank3. Dále specifikujeme, že každý z nich chceme popsat MODELem MixingTank - Tank1, Tank2 a Tank3 jsou *případy* MODELu MixingTank.

V sekci EQUATION MODELu MixingTankCascade jsou zadány rovnice, které specifikují spojení tří nádob zadáním rovnosti průtoku a složení na odpovídajících vstupech a výstupech. Poznamenejme, že v MODELu vyšší úrovně je možno použít všech proměnných obsažených v subMODELech nižší úrovně, z nichž se sám skládá. K odkazu na proměnné určitého případu se používá tečkové notace zavedené v odstavci 2.3.2.

Další příklad spojování modelů je uveden v dodatku A.2, kde je ukázán vstupní soubor pro příklad mísiče s regulací hladiny.

2.4.2 Proudy - STREAMs

Alternativně k výše ukázanému přístupu ke spojování modelů může být použito mechanismu nazývaného STREAM. Ten dovoluje propojit MODELy s jinými MODELy efektivním a koncepčně jasným způsobem. Jak uvidíme dále, jsou proudy - STREAMy - pouze podmnožinami proměnných MODELu. Použijme konceptu STREAM na příklad kaskády míchacích nádob.

Stejně jako pro proměnné, musí být typy proudů - STREAMů nejdříve být deklarovány v sekci DECLARE.

```
DECLARE

  TYPE
    Mole          =   20  :     0  :  1E3  UNIT = "kmole"
    MoleFlowrate  =  1.0  :     0  :  1E3  UNIT = "kmole/min"
    ...
    MolarFraction =  0.5  :     0  :  1E0  UNIT = "-"
    ...

  STREAM
    MAINSTREAM    IS MoleFlowRate, MolarFraction

END #DECLARE
```

Uvedená deklarace udává, že proud - STREAM s názvem MAINSTREAM obsahuje proměnnou typu MoleFlowRate následovanou proměnnou typu MolarFraction (v tomto pořadí).

Obecně má deklarace proudu - STREAMu- následující podobu:

STREAM

StreamName IS *list of variable types (separated by ",")*

kde *StreamName* je jméno, kterým se budeme na proud - STREAM odkazovat v MODELech a *seznam typů proměnných* je podmnožina dříve deklarovaných typů proměnných.

V rámci MODELu musí být proudy - STREAMy deklarovány v sekci STREAM. Např. v MODELu MixingTank chceme použít proudy - STREAMy - Inlet1,Inlet2 and Outlet, které obsahují proměnné použité ke spojení míchacích nádob v kaskádě.

```
MODEL MixingTank

  PARAMETER
    NoComp              AS INTEGER
    ...
  VARIABLE
    FlowIn1, FlowIn2    AS MoleFlowRate
    FlowOut1, FlowOut2  AS MoleFlowRate
    ...
    xIn1,xIn2,xOut      AS ARRAY (nocomp) OF MolarFraction
    ...
  STREAM
    Inlet1 : FlowIn1, xIn1      AS MAINSTREAM
    Inlet2 : FlowIn2, xIn2      AS MAINSTREAM
    Outlet : FlowOut1,xOut      AS MAINSTREAM

  EQUATION
  ...
```

Tyto proudy zahrnují vstupní a výstupní proměnné průtoků a složení MODELu (FlowIn1, xIn1, FlowIn2,...). Všechny jsou dříve deklarovaného typu MAINSTREAM.

Poznámka 2.6 *Poznamenejme, že deklarace* STREAM *v sekcích* DECLARE *a* MODEL *musejí být konsistentní vzhledem k typům proměnných v nich obsaženýma jejich pořadí.*

Jakmile toto bylo učiněno, vstupní a výstupní proměnné MODELu MixingTank byly spojeny v proudech Inlet1, Inlet2 and Inlet3. V sekci EQUATION MODELu MixingTank se nic nezmění.

Nyní je možno napsat přímo propojovací rovnice MODELu MixingTankCascade.

```
MODEL MixingTankCascade
```

```
  UNIT
    Tank1 AS MixingTank
    Tank2 AS MixingTank
    Tank3 AS MixingTank

  STREAM
    Inlet IS Tank1.Inlet
    Outlet IS Tank3.Outlet

  EQUATION
    Tank2.Inlet1 = Tank1.Outlet;
    Tank3.Inlet1 = Tank2.Outlet;

END  #MODEL MixingTankCascade
```

V sekci `EQUATION` vidíme, že použitím proudů můžeme spojovat různé mísiče jednoduchým zadáním rovnosti odpovídajících si proudů. Výhoda úspory v zápisu `MODEL`u je zřejmější v případě, že proudy - `STREAM`s - obsahují mnoho proměnných. Avšak podobně jako v uvedeném příkladě je používání proudů doporučeníhodné, protože většinou vede k lepší koncepční struktuře a tím také k lepší čtivosti `MODEL`ů.

Navíc k proudům v `MODEL`u `MixingTank` deklarujeme rovněž nové proudy v sekci `STREAM MODEL`u `MixingTankCascade`. Podle této deklarace se můžeme odkázat na vstup do míchací nádoby `Tank1` se jménem `Inlet` a na výstup (který je výstupem z `Tank3`) pomocí `Outlet`. Vně modelu se budeme např. odkazovat na proud `Inlet` pomocí:

instance name of model MixingTankCascade.`Inlet`

Poznamenejme, že notace použitá pro deklaraci proudů v `MODEL`u `MixingTankCascade` je zkratkou pro:

```
STREAM
    Inlet  : Tank1.FlowIn1, Tank1.xIn1 AS MAINSTREAM
    Outlet : Tank3.FlowOut, Tank3.xOut AS MAINSTREAM
```

Jak bylo uvedeno dříve, jsou `STREAM`s v *gPROMS* pouze zkratkami pro odkazy určité sady proměnných. Proto mohou být použity pro proměnné jakéhokoliv typu, které by měly být součástí rozhraní `STREAM` v `MODEL`u. Dokonce i když proudy - `STREAM`s mohou často odpovídat hmotným proudům mezi různými procesními jednotkami v provozu, měli bychom je chápat jako *proudy informací*, které mohou obsahovat jakoukoliv informaci.

2.5 Distribuované modely

Reaktor s pístovým tokem zmíněný v kapitole 1.3.2 je popsán pomocí proměnných, které závisejí na čase a axiální souřadnici uvnitř reaktoru z. K deklaraci takového *distribuovaného modelu* musí být zavedena přídavná sekce `DISTRIBUTION_DOMAIN`, sekce `VARIABLE` a `EQUATION` musejí být také upraveny. Sekce `MODEL` pro reaktor s pístovým tokem pak bude v následujícím tvaru

```
MODEL MPlugFlowReaction

  PARAMETER
    TubeLength  AS REAL

  DISTRIBUTION_DOMAIN
    Axial       AS [ 0 : TubeLength ]

  VARIABLE
    c           AS DISTRIBUTION(Axial) OF concentration
    v           AS velocity
    k           AS reacconst

  BOUNDARY
    # @ z = 0
        c(0) = 2 ;

  EQUATION

# Mass balance including reaction term
  PARTIAL(c(0|+:TubeLength),Axial) + 1/v*$c(0|+:TubeLength) =
                                        - k * c(0|+:TubeLength);

END # Model MPlugFlowReaction
```

Pro každou distribuovanou proměnnou musí být specifikován interval, ve kterém proměnná závisí na distribuční souřadnici. Tyto tzv. distribuční obory - *distribution domains* - jsou deklarovány v sekci `DISTRIBUTION_DOMAIN`. V sekci `DISTRIBUTION_DOMAIN` `MODELu MPlugFlowReaction` je délková souřadnice deklarována jako distribuční obor s názvem `Axial` v rozmezí od 0 do délky specifikované parametrem `TubeLength`. Pro účely rozlišení mezi vnitřkem a hranicemi distribučního oboru se používá následujícího zápisu:

V sekci `VARIABLE MODELu MPlugFlowReaction` je proměnná `c` deklarována v distribuční oblasti `Axial`, přičemž je typu `concentration`. Teď tato proměnná může být použita v v sekci `EQUATION`. Pro účely odkazu na distribuovanou proměnnou v

Tabulka 2: Zápis distribučního oboru

Matematická oblast	zápis v *gPROMS*
[a,b]	a : b
(a,b]	a|+ : b
[a,b)	a : b|-
(a,b)	a|- : b|-
[a,b/2]	a : b/2
[a,b/2)	a : b/2|-

určité podoblasti distribuční oblasti může být toto deklarováno v závorkách za identifikátorem proměnné. Např.

```
c(0:Tubelength/2)
```

se vztahuje k `c` mezi `0` a `TubeLength/2` včetně hranic, zatímco

```
c(0|+:Tubelength/2|-)
```

se vztahuje k `c` mezi `0` a `TubeLength/2` bez hranic. Odkaz na `c` v bodě `z=0` se píše jako

```
c(0).
```

Okrajové podmínky jsou součástí popisu fyzikálního chování systému. Proto se k nim přistupuje jako ke každé jiné rovnici zadané v sekci `EQUATION`. Alternativou k zadání okrajových podmínek je jejich zadání ve zvláštní sekci s klíčovým slovem `BOUNDARY`. To zajistí lepší čitelnost `MODELu`.

V sekci `EQUATION MODELu MPlugFlowReaction` je zadána PDE. Parciální diferenciace vzhledem k prostorové souřadnici se zadává pomocí operátoru `PARTIAL`:

`PARTIAL` (*Expression*, *DistributionDomain*)

kde *Expression* je diferencovaný člen vzhledem *DistributionDomain*. *DistributionDomain* je distribuční oblast deklarovaná dříve. Následují příklady prvních parciálních derivací:

$\frac{\partial T}{\partial z}$	`PARTIAL(T,Axial)`
$\frac{\partial (uC)}{\partial r}$	`PARTIAL(u*C,Radial)`
$\frac{\partial T}{\partial t}$	$T

Nakonec je třeba v sekci `PROCESS SET` zadat prostorovou diskretizaci. V našem příkladě je to uděláno pomocí:

```
PROCESS plug

  ....

  SET # Parameter values
    WITHIN PlugFlowReaction DO
      ...
      Axial := [CFDM,2,40] ;
    END # Within PlugFlowReaction
```

Tento příkaz znamená, že `Axialní` souřadnice bude diskretizována pomocí schématu centrálních konečných diferencí se 40 diskretizačními intervaly, kde aproximace derivací je druhého řádu.

Poznámka 2.7 *Detailní popis dostupných diskretizačních metod je uveden v gPROMS Introductory User Guide .*

3 Použití *gPROMS*

3.1 Spuštění gPROMS

3.1.1 Vytvoření pracovního adresáře

Před spuštěním *gPROMS* byste měli založit pracovní adresář pro *gPROMS*. Poté byste měli založit podadresář pro své vstupní soubory se jménem `input`. Po svém spuštění zde *gPROMS* hledá vstupní soubory.

3.1.2 Start *gPROMS*

gPROMS se spouští provedením příkazu `gproms`. Jestliže je *gPROMS* správně nainstalován, spustí se ve zvláštním okně program *gRMS*, který se používá pro vizualizaci dat a systém odpoví (na počítačích SUN):

```
gPROMS - Version 1.8.0 for SunOS 5.7  Jun 26 2000
general PROcess Modelling System

Copyright 1995-1997 Imperial College of Science,
Technology and Medicine
```

```
        Copyright 1997-2000 Process Systems Enterprise Limited
        All rights reserved. This software program is protected
        by UK and international laws. Unauthorized reproduction
        and utilization is strictly forbidden. Refer to the software
        licence agreement for acceptable policies.

        Enter HELP for a summary of available commands.
        Enter NEW to display new features of a gPROMS release.
        Visit WWW at http://www.psenterprise.com/ for more information.
        E-mail support.gPROMS@psenterprise.com for product support.

    gPROMS>>
```

Nyní můžete používat příkazů *gPROMS* pro výběr a spuštění simulací.

3.2 Příkazy *gPROMS*

dir Tento příkaz ukáže všechny dostupné vstupní soubory ve vstupním adresáři. Příklad:

```
gPROMS>> dir
The following input files are available:

control.gPROMS  plug.gPROMS  tank.gPROMS

gPROMS>>
```

select Příkaz se používá k výběru jednoho ze vstupních souborů pro simulaci, která Vás zajímá.

```
gPROMS>> select tank


Translating file : tank


gPROMS translation initialization took 0.001 seconds.

 .. MODEL MIXINGTANK
 .. PROCESS MIX1

gPROMS translation took 0.107 seconds.

The following processes are available:
```

```
  MIX1

gPROMS>>
```

execute Provedení tohoto příkazu spustí aktuální simulaci podle zadání specifikovaného v sekci PROCESS.

```
gPROMS>> execute mix1

Executing process MIX1...

All 20 variables will be monitored during this simulation!

Building mathematical problem description took 0.004 seconds.

Loaded MA48 library
Execution begins....

   Variables
       Known            :      9
       Unknown          :     11
          Differential  :      2
          Algebraic     :      9

   Model equations      :     11
   Initial conditions   :      2


Checking consistency of model equations and
ASSIGN specifications... OK!

Checking index of differential-algebraic equations
(DAEs)...         OK!

Checking consistency of initial conditions...
                        OK!
Solving block 5 (out of 8) involving
              6 variables and equations ...
     Initialization calculation compl
  CONTINUE FOR 20 executed at 0.000
Integrating from 0.000 to 2.000
 ......
Integrating from 100.000 to 102.000
```

```
Time event occurs at 100.000

Execution of MIX1 completed successfully.

gPROMS simulation took 0.239 seconds.

 Total CPU Time:        0.200 seconds.
  User CPU Time:        0.130 seconds.
System CPU Time:        0.070 seconds.

gPROMS>>
```

write Zapíše rozšířenou formulaci problému do souboru `gPROMS.lst` v pracovním adresáři *gPROMS*. Prostudování tohoto souboru může být velmi užitečné při hledání a odstraňování chyb.

end Ukončuje Vaši práci s *gPROMS*. (Poznámka: *gRMS* musí být ukončen zvlášť.)

Poznámka 3.1 *Abychom se vyhnuli psaní dlouhých příkazů, stačí psát jen tolik písmen, aby byl příkaz odlišitelný od ostatních:*

Příklad: "`gPROMS>> se tank`*" je stejné jako "*`gPROMS>> select tank`*".*

3.3 Grafická analýza výsledků pomocí *gRMS*

gRMS je zvláštní program, který dostává data produkovaná pomocí *gPROMS* pro kreslení a tisk 2D a 3D grafů. Při prohlížení uloženého procesu je možno vybrat každou proměnnou, která byla specifikována v sekci `MONITOR` pro simulační výpočet.

Standardními grafy jsou časové průběhy skalární proměnné v grafu 2D. 3D grafy jsou užitečné k vizualizaci řešení distribuovaných modelů.

Standardní použití *gRMS* je přímočaré a proto zde není popisováno.

Poznámka 3.2 *Detailní popis všech funkcí lze nalézt v gPROMS Introductory User Guide appendix A.*

A Příklady vstupních souborů

V tomto dokumentu jsme často ukazovali části zdrojového kódu, abychom vysvětlili určitou syntaxi. Úplné vstupní soubory pro *gPROMS*, ze kterého pocházely tyto příklady, jsou uvedeny v tomto dodatku. Odpovídající soubory by měly být umístěny ve vašem vstupním - `input` adresáři.

`tank.gPROMS`: Zdrojový kód odpovídající příkladu uvedenému v sekci 1.3.1.

`control.gPROMS`: Toto je model nádoby s přidaným regulátorem hladiny.

`plug.gPROMS`: Zdrojový kód odpovídající příkladu v sekci 1.3.2.

A.1 Míchaná nádoba s přepadem (tank.gPROMS)

```
{
  Simple Mixing tank model (only material balance) with overflow
  weir.
  Course material for gPROMS course on dynamic process simulation
  (example 1):

  Mixing of two streams (Water & Ethanol):

                ------------
               |            |<-- FlowIn1 (Water)
               |            |<-- FlowIn2 (Ethanol)
               |            |
   FlowOut2 <--  |          |
                 |----------|     -
                 |     N    |   Level
                  ----------      -
                     |
                     |--> FlowOut1

Revision History:
  v. 1.0:  10.07.2000 - Heiko Briesen (LPT)
}

DECLARE

  TYPE
    Mole          =   20  :     0  :  1E3  UNIT = "kmole"
    MoleFlowrate  =  1.0  :     0  :  1E3  UNIT = "kmole/min"
    Length        =  1.0  :     0  :   10  UNIT = "m"
    Volume        =  1.0  :     0  :   50  UNIT = "m3"
    MolarFraction =  0.5  :     0  :  1E0  UNIT = "-"
    MolarDensity  =   50  :     0  :  100  UNIT = "kmol/m3"

  STREAM
    MAINSTREAM    IS MoleFlowRate, MolarFraction
```

```
END

MODEL MixingTank

  PARAMETER
    NoComp              AS INTEGER
    Area                AS REAL
    WeirHeight          AS REAL
    Alpha               AS REAL

  VARIABLE
    FlowIn1, FlowIn2    AS MoleFlowRate
    FlowOut1, FlowOut2  AS MoleFlowRate
    HoldUp              AS ARRAY (nocomp) OF Mole
    xIn1,xIn2           AS ARRAY (nocomp) OF MolarFraction
    xOut1,xOut2,xTank   AS ARRAY (nocomp) OF MolarFraction
    TotalHoldUp         AS Mole
    RhoMolComp          AS ARRAY (nocomp) OF MolarDensity
    RhoHelp             AS ARRAY (nocomp) OF MolarDensity
    RhoMol              AS MolarDensity
    Vol                 AS Volume
    Level               AS Length

  STREAM
    Inlet1 : FlowIn1, xIn1      AS MAINSTREAM
    Inlet2 : FlowIn2, xIn2      AS MAINSTREAM
    Outlet1: FlowOut1,xOut1     AS MAINSTREAM
    Outlet2: FlowOut2,xOut1     AS MAINSTREAM

  EQUATION

    # Mass balance
    $HoldUp = FlowIn1 * xIn1 + FlowIn2 * xIn2 - ( FlowOut1 ) * xOut1
                                        - ( FlowOut2 ) * xOut2;

    # molar concentrations
    # Example for the FOR construct
    FOR i := 1 TO NoComp DO
        Holdup(i) = TotalHoldUp * xTank(i) ;
    END

    TotalHoldUp = SIGMA(HoldUp);

    # Level
```

```
    Vol = Level * Area;
    Vol * RhoMol = TotalHoldUp;

    # well-mixedness in tank
    xTank = xOut1;
    xTank = xOut2;

    # Overflow over weir
    IF Level > WeirHeight THEN
         FlowOut2 = Alpha * SQRT(Level - WeirHeight);
    ELSE
         FlowOut2 = 0;
    END # IF

    # Molar Density of mixture
    RhoHelp = xTank / RhoMolComp;
    RhoMol = 1 / SIGMA(RhoHelp);

END # Model Tank

PROCESS mix1

  UNIT
    Mixer1 AS MixingTank

  MONITOR
       Mixer1.FlowOut1;
       Mixer1.FlowOut2;
       Mixer1.xTank(*);
       Mixer1.Level;

  SET # Parameter values
    WITHIN Mixer1 DO
        NoComp := 2;
    END # WITHIN Mixer1
  # Alternative with complete pathname notation
    Mixer1.Area := 1;
    Mixer1.WeirHeight := 2;
    Mixer1.Alpha := 1;

  ASSIGN # Degrees of freedom
    WITHIN Mixer1 DO
        FlowIn1 :=10;
        xIn1(1) :=1;
```

```
        xIn1(2) :=0;
        FlowIn2 :=10;
        xIn2(1) :=0;
        xIn2(2) :=1;
        FlowOut1 :=20;
        RhoMolComp(1) := 55.39; # Water
        RhoMolComp(2) := 17.15; # Ethanol
    END # WITHIN Mixer1

  INITIAL # Initial conditions
    WITHIN Mixer1 DO
         xTank(1) = 1;
         TotalHoldUp = 55.39;
    END # WITHIN Mixer1

  SOLUTIONPARAMETERS
     ReportingInterval  := 2;

  SCHEDULE # Operating procedure
     SEQUENCE
        CONTINUE FOR 20
        RESET
           Mixer1.FlowIn1 := 12;
        END
        CONTINUE FOR 20
        RESET
           Mixer1.FlowIn1 := 10;
        END
        CONTINUE FOR 20
        RESET
           Mixer1.FlowIn1 := 8;
        END
        CONTINUE FOR 20
        RESET
           Mixer1.FlowIn1 := 10;
        END
        CONTINUE FOR 20
     END # SEQUENCE

END # mix1
```

A.2 Míchaná nádoba s přepadem a regulátorem hladiny (control.gPROMS)

```
{
  Simple Mixing tank model (only material balance) with overflow
  weir and level control.
  Course material for gPROMS course on dynamic process simulation
  (example 2):

  Mixing of two streams (Water & Ethanol):

                ------------
               |            |<-- FlowIn1 (Water)
               |            |<-- FlowIn2 (Ethanol)
               |            |
   FlowOut2 <--  |          |
                 |----------| ----- LevCont
                 |    N     |   Level  |
                  ----------      -    |
                     |                 |
                     |---------------------> FlowOut1

Revision History:
  v. 1.0:  10.07.2000 - Heiko Briesen (LPT)
}

DECLARE

  TYPE
    Mole          =   20  :     0  :  1E3  UNIT = "kmole"
    MoleFlowrate  =  1.0  :     0  :  1E3  UNIT = "kmole/min"
    Length        =  1.0  :     0  :   10  UNIT = "m"
    Volume        =  1.0  :   0.0  :   50  UNIT = "m3"
    MolarFraction =  0.5  :     0  :  1E0  UNIT = "-"
    MolarDensity  =   50  :     0  :  100  UNIT = "kmol/m3"
    NoType        =  1.0  :  -1E5  :  1E5  UNIT = "-"

  STREAM
    MAINSTREAM    IS MoleFlowRate, MolarFraction

END

MODEL MPI_Cont
```

```
  PARAMETER
    clip        AS REAL              # output unclipped
    normal      AS REAL              # output unscaled/unnormalised

  VARIABLE
#   Connection:
      I_in      AS   notype
      SetPoint  AS   notype
      I_out     AS   notype
#   Internal:
      bias      AS   notype          # ss control value
      cont      AS   notype          #
      error     AS   notype          # setpnt & variable error
      gain      AS   notype          # controller gain
      I_error   AS   notype          # integral of error
      min       AS   notype          # min. scaled output value
      max       AS   notype          # max. scaled output value
      resetval  AS   notype          # integral time
      value     AS   notype          # calculated value

  EQUATION

    error    = SetPoint - I_in ;
    $I_error = error ;
    value    = bias + gain*( error + I_error/resetval ) ;

#   Clip if required
      IF clip = 1 THEN
        IF value > max THEN
          cont = max;
        ELSE
                IF value < min THEN
                        cont = min;
                ELSE
                        cont = value;
                END
        END
      ELSE
        cont = value;
      END

#   Scale if required
      IF normal > 0 THEN
        I_out = normal*( cont - min )/( max - min );
```

```
      ELSE
        I_out = cont;
      END


END # MPI_Cont

MODEL MixingTank

 <same as in the previous example>

END # Model Tank

MODEL MixerWithControl
  UNIT
    Tank                AS MixingTank
    LevelControl        AS MPI_Cont
  EQUATION
# Level Control
    Tank.FlowOut1 = LevelControl.I_out;
    Tank.Level =  LevelControl.I_in;

END # Model TankWithControl

PROCESS mix2

  UNIT
    Mixer2 AS MixerWithControl

  SET # Parameter values
    WITHIN Mixer2.Tank DO
        NoComp := 2;
        Area := 1;
        WeirHeight := 2;
        Alpha := 1;
    END # WITHIN Mixer2.Tank
    WITHIN Mixer2.LevelControl DO
        Clip := 1;
        normal := 0;
    END # WITHIN Mixer2.LevelControl

  ASSIGN # Degrees of freedom
    WITHIN Mixer2.Tank DO
        FlowIn1 :=10;
```

```
        xIn1(1) :=1;
        xIn1(2) :=0;
        FlowIn2 :=10;
        xIn2(1) :=0;
        xIn2(2) :=1;
#       FlowOut1 :=20;
        RhoMolComp(1) := 55.39; # Water
        RhoMolComp(2) := 17.15; # Ethanol
    END # WITHIN Mixer2.Tank
    WITHIN Mixer2.LevelControl DO
        gain  :=-100 ;
        resetval := 1;
        SetPoint    := 1;
        MIN   := 0;
        MAX   := 10000;
        BIAS :=  20;
    END # WITHIN Mixer2.LevelControl

  INITIAL # Initial conditions
    WITHIN Mixer2.Tank DO
         xTank(1) = 1;
         TotalHoldUp = 55.39;
    END # WITHIN Mixer2.Tank
    WITHIN Mixer2.LevelControl DO
         I_ERROR = 0;
    END # WITHIN Mixer2.LevelControl


  SOLUTIONPARAMETERS
     BlockDecomposition   := ON ;
     Integrator           := "SRADAU" ;
     LASolver             := "MA48" ;
     OutputLevel          := -1 ;
     ReportingInterval    :=  2;

  SCHEDULE # Operating procedure
     SEQUENCE
        CONTINUE FOR 20
        RESET
           Mixer2.Tank.FlowIn1 := 12;
        END
        CONTINUE FOR 20
        RESET
           Mixer2.Tank.FlowIn1 := 10;
```

```
        END
        CONTINUE FOR 20
        RESET
           Mixer2.Tank.FlowIn1 := 8;
        END
        CONTINUE FOR 20
        RESET
           Mixer2.Tank.FlowIn1 := 10;
        END
        CONTINUE FOR 20
     END # SEQUENCE
END # mix2
```

A.3 Model reaktoru s pístovým tokem (plug.gPROMS)

```
{
  Simple plug flow reactor to illustrate the handling of distributed
  models.
  Course material for gPROMS course on dynamic process simulation
  (example 3):

Revision History:
  v. 1.0:  10.07.2000 - Heiko Briesen (LPT)
}

DECLARE
  TYPE
    velocity         =  1.0  :  -1E5  :  1E5  UNIT = "m/s"
    Concentration    =  1.0  :  -1E5  :  1E5  UNIT = "kmol/m3"
    ReacConst        =  1.0  :  -1E5  :  1E5  UNIT = "1/s"
END

MODEL MPlugFlowReaction

  PARAMETER
    TubeLength  AS REAL

  DISTRIBUTION_DOMAIN
    Axial       AS [ 0 : TubeLength ]

  VARIABLE
    c           AS DISTRIBUTION(Axial) OF concentration
    v           AS velocity
```

```
    k           AS reacconst

  BOUNDARY
    # @ z = 0
        c(0) = 2 ;

  EQUATION
# PDE for numerical solution
  PARTIAL(c(0|+:TubeLength),Axial) + 1/v*$c(0|+:TubeLength) =
                                        - k * c(0|+:TubeLength);

END # Model MPlugFlowReaction

PROCESS plug

  UNIT
    PlugFlowReaction AS MPlugFlowReaction

  SET # Parameter values
    WITHIN PlugFlowReaction DO
      TubeLength := 1 ; # m
      Axial := [CFDM,2,40] ;
    END # Within PlugFlowReaction

  ASSIGN # Degrees of freedom
    WITHIN PlugFlowReaction DO
      v := 1 ;
      k := -1 ;
    END # Within PlugFlowReaction

  INITIAL # Initial conditions
    WITHIN PlugFlowReaction DO
      FOR z := 0|+ TO TubeLength DO
            c(z) = 2 ;
      END # For z
    END # Within PlugFlowReaction

  SOLUTIONPARAMETERS
     ReportingInterval  := 0.1;

  SCHEDULE # Operating procedure
    CONTINUE FOR 1.5

END # Process plug
```