POLITECHNIKA ŚLĄSKA

WYDZIAŁ AUTOMATYKI, ELEKTRONIKI I INFORMATYKI

KIERUNEK AUTOMATYKA I ROBOTYKA

Projekt inżynierski

Program w LabVIEW dla określenia charakterystyk roboczych procesu fermentacji z opóźnionym hamowaniem produktem

Autor: Marcin Biczyski

Kierujący pracą: Prof. dr hab. inż. Mieczysław Metzger

Gliwice, styczeń 2017

Spis treści

Spis treści **3**

1 Wstęp **5**

2 Obiekt symulacji **5**

2.1 Model matematyczny 6

2.2 Model zdyskretyzowany 7

2.3 Równania obliczeń numerycznych 7

3 Opis symulatora **7**

3.1 Panel przedni 8

3.2 Diagram blokowy 14

3.2.1 Pętla główna obiektu 14

3.2.2 Metoda UDYN 15

3.2.3 Realizacja opóźnienia 16

3.2.4 Określanie charakterystyk roboczych 18

3.2.5 Zapis do pliku 18

4 Przykładowe wyniki symulacji **19**

5 Podsumowanie **21**

Bibliografia **22**

Skrypt *CFPlplot.m* **23**

Spis rysunków **24**

1 Wstęp

Jedną z nieodzownych części projektowania i wdrażania procesu przemysłowego jest dokładna symulacja komputerowa. Pozwala ona w większym lub mniejszym stopniu odtworzyć proces na podstawie równań matematycznych opisujących go. Rozwiązując numerycznie kolejne równania można określić przybliżony stan układu w dowolnym momencie. Jest to jedna z metod badania obiektów nieliniowych, dla których analityczne rozwiązywanie modelu jest znacznie utrudnione lub wręcz niemożliwe.

Dodatkowymi zaletami symulacji jest możliwość szybkiej analizy procesu rzeczywistego, którego obserwacja zajęłaby wiele dni, miesięcy, czy lat. Pozwala to w znacznym stopniu zredukować koszty, a także zwalnia z potrzeby budowy instalacji laboratoryjnych.

Przedmiotem pracy było umożliwienie określenia charakterystyk roboczych procesu fermentacji z opóźnionym hamowaniem produktem w zadanym punkcie pracy. Aby to zrealizować, stworzono w środowisku graficznego programowania LabVIEW program pozwalający na długą symulację procesu ([1]) poprzez numeryczne rozwiązywanie układu równań i przedstawienie wyników w postaci wykresów.

Zakres pracy obejmował implementację równań rozpatrywanego obiektu, zaproponowanie i implementację metody realizacji opóźnienia, umożliwienie symulacji dla zmieniających się parametrów oraz stworzenie przejrzystego i intuicyjnego interfejsu użytkownika. Program umożliwia użytkownikowi ręczny lub automatyczny dobór przedziałów parametrów, dla których generowane są charakterystyki, a także zapis danych do pliku w celu dalszej obróbki. Dodatkowo, utworzono skrypt programu MATLAB pozwalający na wyświetlanie trójwymiarowych wykresów badanych charakterystyk.

2 Obiekt symulacji

W warunkach przemysłowych etanol jest często uzyskiwany za pomocą drożdży *Saccharomyces cerevisiae* czy bakterii *Zymomonas mobilis* w procesie ciągłej fermentacji. Niestety, proces ten jest skomplikowany i silnie nieliniowy. Dla pewnych wartości parametrów występują niegasnące oscylacje w kluczowych zmiennych procesowych (produkcie i biomasie). Jak pokazano w [2], zachowanie takie powstaje tylko przy udziale czynnika hamującego, który może być związany z opóźnioną reakcją mikroorganizmów na zmiany w środowisku ([3]).

Przedmiotem symulacji jest jednosubstratowy bioreaktor realizujący proces fermentacji z

opóźnionym hamowaniem produktem. Z powodu wysokiej nieliniowości procesu do wyznaczania charakterystyk roboczych i poszukiwania niegasnących oscylacji zostanie wykorzystana metoda wielokrotnej symulacji ([4]).

2.1 Model matematyczny

W celu wyprowadzenia matematycznego modelu procesu należy przyjąć następujące założenia:

- Całość mikroorganizmów w bioreaktorze (biomasa) opisana jest pojedynczą zmienną X
- W bioreaktorze występuje mieszanie idealne, a jego objętość jest stała ($V = const$)
- Mikroorganizmy nie wymagają dodatkowe podtrzymywania funkcji życiowych

Na tej podstawie, z bilansu masy substratu, biomasy i produktu, a także równania Luedekinga-Pireta ([5]) można wyznaczyć równania stanu nieustalonego:

$$\frac{dS}{dt} = D(S_{in} - S) - \frac{\mu(S, P_\tau)}{Y} * X \tag{1}$$

$$\frac{dX}{dt} = -DX + \mu(S, P_\tau) * X \tag{2}$$

$$\frac{dP}{dt} = -DP - \mu(S, P_\tau) Y_P * X \tag{3}$$

gdzie: S_{in}, S - stężenie substratu wejściowego i wyjściowego [g/L]; X - stężenie biomasy [g/L]; D - szybkość rozcieńczania (ang. *dilution rate*) [$1/h$]; Y - współczynnik wzrostu biomasy (ang. *biomass yield coefficient*) - ilość biomasy wytworzonej z 1 grama skonsumowanego substratu [g/g]; Y_P - współczynnik wzrostu produktu (ang. *product yield coefficient*) - ilość produktu wytworzonego z 1 grama skonsumowanego substratu [g/g].

Właściwa szybkość wzrostu (ang. *specific growth rate*) $\mu(S, P_\tau) * X$ opisana jest następującą zależnością:

$$\mu(S, P_\tau) = \frac{\mu_m S}{S + K_S} * \frac{K_i}{P_\tau + K_i}; \qquad P_\tau := P(t - \tau) \tag{4}$$

gdzie: μ_m - maksymalna Właściwa szybkość wzrostu [$1/h$]; K_S - stała pół-nasycenia (ang. *half-saturation constant*, inna nazwa: stała Monoda) [g/L]; K_i - stała hamowania (ang. *inhibitory constatnt*) [g/L]. Pierwsza część równania (4) opisuje model dobrze znany model Monoda, natomiast druga dotyczy opóźnionego hamowania produktem.

2.2 Model zdyskretyzowany

W celu umożliwienia numerycznego rozwiązania ciągłych równań stanu nieustalonego należy je wcześniej aproksymować równaniami dyskretnymi:

$$S_{k+1} = D(S_{in_k} - S_k) - \frac{\mu(S_k, P_{\tau_k})}{Y} * X_k \tag{5}$$

$$X_{k+1} = -DX_k + \mu(S_k, P_{\tau_k}) * X_k \tag{6}$$

$$P_{k+1} = -DP_k - \mu(S_k, P_{\tau_k})Y_P * X_k \tag{7}$$

$$\mu(S_k, P_{\tau_k}) = \frac{\mu_m S_k}{S_k + K_S} * \frac{K_i}{P_{\tau_k} + K_i} \tag{8}$$

Dalsze wyprowadzenie dla zmiennej P_{τ_k} opisane zostało w sekcji 3.2.3 wraz z dwiema propozycjami implementacji w kodzie programu.

2.3 Równania obliczeń numerycznych

Aby możliwe było wyznaczenie konkretnych wartości parametrów w danej chwili należy numerycznie scałkować równania różnicowe. W tym celu została zastosowana prosta metoda jednokrokowa - metoda Eulera:

$$x_{k+1} = x_k + F_k * h \tag{9}$$

$$F_k = f(x_k, p_k, u_k) \tag{10}$$

gdzie h jest przedziałem czasu pomiędzy kolejnymi iteracjami.

Końcowe równania mają następującą postać:

$$S_{k+1} = S_k + \left[D(S_{in_k} - S_k) - \frac{\mu(S_k, P_{\tau_k})}{Y} * X_k\right] * h \tag{11}$$

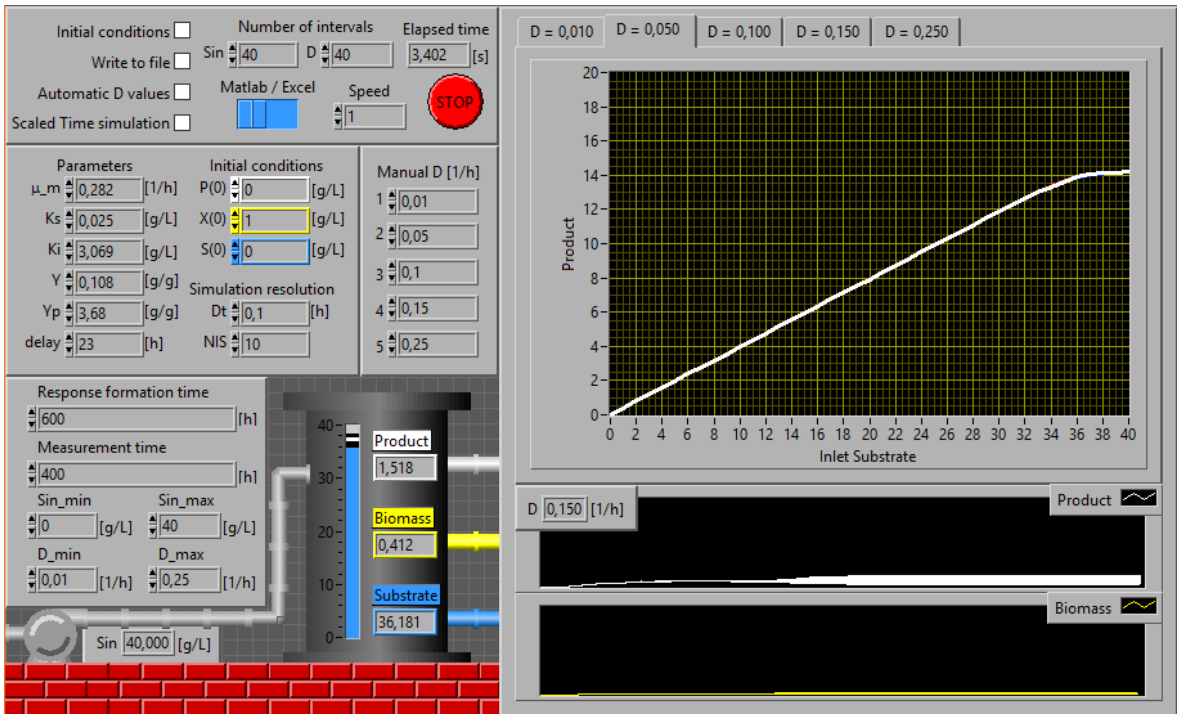
$$X_{k+1} = X_k + \left[- DX_k + \mu(S_k, P_{\tau_k}) * X_k\right] * h \tag{12}$$

$$P_{k+1} = P_k + \left[- DP_k - \mu(S_k, P_{\tau_k})Y_P * X_k\right] * h \tag{13}$$

3 Opis symulatora

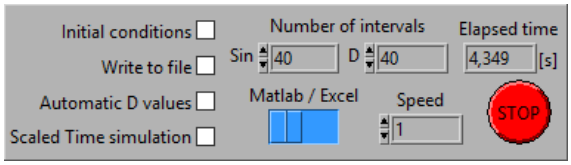
Oprogramowanie symulatora składa się z pliku *CFPI.vi* (ang. *Continous Fermentation Process with Inhibition*), który dzieli się na 2 części: panel przedni (ang. *Front Panel*) z interfejsem użytkownika w języku angielskim oraz diagramu blokowego (ang. *Block Diagram*) realizującego logikę symulacji.

3.1 Panel przedni



Rysunek 1: Panel przedni programu *CFPI.vi*

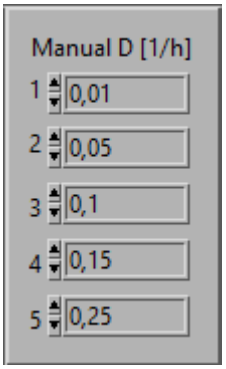
Panel przedni można podzielić na 2 zasadnicze sekcje: ustawień po lewej stronie oraz przebiegów po stronie prawej stronie ekranu. W sekcji ustawień użytkownik może dostosować parametry procesu, przedziały zmienności danych, skonfigurować zapis do pliku, czy określić dokładność i czas symulacji. W trakcie działania programu, w tej sekcji można podejrzeć aktualne parametry symulowanego obiektu. Sekcja przebiegów składa się z wykresów - 2 na dole dotyczących aktualnie symulowanego procesu oraz kolejnych 5 w zakładkach przedstawiających wyniki symulacji w postaci charakterystyk roboczych $P = f(S_{in})|_{D=const}$.



Rysunek 2: Podpanel ustawień trybu pracy symulatora

Najważniejszym elementem podpanelu ustawień trybu pracy symulatora (Rysunek 2) jest czerwony przycisk **STOP**. Pozwala on zatrzymać aktualnie trwającą symulację w razie napotkania nieprzewidzianego zachowania, błędu lub podania nieprawidłowych parametrów. Przy użyciu przełącznika *Initial conditions* po lewej stronie podpanelu użytkownik może zdecydować, czy symulacja będzie rozpoczynać się z podanych przez niego warunków początkowych, czy z domyślnych wartości ($P(t=0)=0; X(t=0)=1; S(t=0)=0[g/L]$).

Przełącznik *Write to file* pozwala określić, czy dane symulacji mają zostać zapisane na komputerze. Po wybraniu tej opcji użytkownik może zadecydować, czy powstały plik ma być sformatowany pod kątem współpracy z programem MATLAB czy programem Microsoft Excel za pomocą przełącznika *Matlab/Excel*. W zależności od ustawień, pok zakończeniu symulacji, w katalogu, w którym znajduje się program powstanie plik CSV *data_matlab.csv* lub *data_excel.csv*.



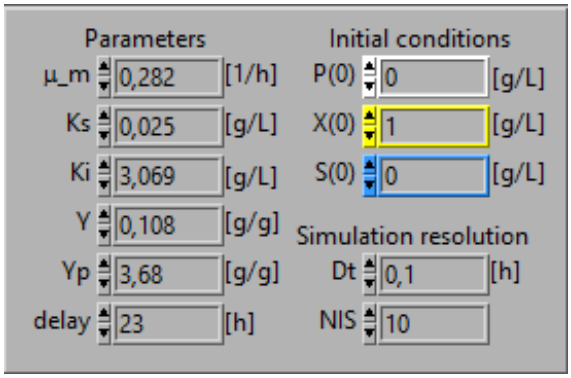
Rysunek 3: Podpanel ręcznych wartości szybkości rozcieńczania

Za pomocą przełącznika *Automatic D values* użytkownik może określić sposób doboru wartości szybkości rozcieńczania - ręczny lub automatyczny. Domyślnym trybem jest tryb ręczny, który umożliwia przeprowadzenie symulacji dla 5 dowolnie wybranych wartości parametru D wprowadzonnych w podpanelu pokazanym na rysunku 3. W przypadku wybrania trybu automatycznego należy określić pożądane punktu początkowe D_{min} i D_{max} (Rysunek 5), a także pożądaną liczbę przedziałów wartości zmiennej (ang. *Number of intervals*) w podpanelu ustawień trybu pracy symulatora. Należy zauważyć, że przy ustawionej liczbie przedziałów n, symulacja zostanie przeprowadzona dla $n + 1$ wartości D. W taki sam sposób można wybrać liczbę przedziałów wejściowego substratu S_{in}.

Przełącznik *Scaled Time simulation* umożliwia przeprowadzenie symulacji z zachowaniem określonej rozdzielczości czasowej. Zastosowana skala to $1 : 3600$, co oznacza że 1 sekunda czasu rzeczywistego odpowiada 3600 sekundom (1 godzinie) czasu symulacji. Skalę tą można modyfikować za pomocą pola *Speed*, w którym można ustawić dodatkowy mnożnik skali. Dla przykładu, przy wartości *Speed* równej 10, skala czasowa wynosi $1 : 36000$, a więc jedna sekunda czasu rzeczywistego odpowiada 10 godzinom symulacji. W przypadku, gdy przełącznik *Scaled Time simulation* jest odznaczony, symulacja wykonuje się z maksymalną możliwą prędkością.

Ostatnim polem podpanelu ustawień trybu pracy jest *Elapsed time*, które pokazuje liczbę sekund czasu rzeczywistego, które upłynęły od rozpoczęcia symulacji lub łączny czas rzeczywisty symulacji, w przypadku jej zakończenia.

Podpanel parametrów procesu jest wykorzystywany do określenia stałych wartości opisujących symulowany proces. Za jego pomocą można określić wartości μ_m, K_S, K_i, Y oraz Y_P (oznaczenia pokrywają się z zastosowanymi przy opisie modelu matematycznego). Parametr delay oznacza opóźnienie τ wykorzystywane w $P_\tau = P(t - \tau)$.

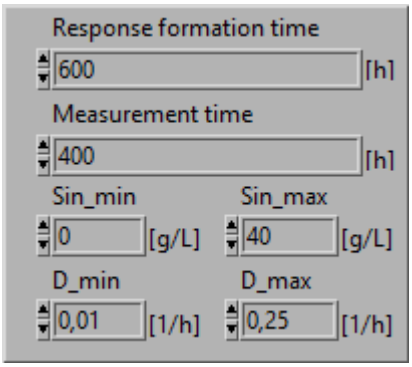


Rysunek 4: Podpanel ustawień parametrów procesu

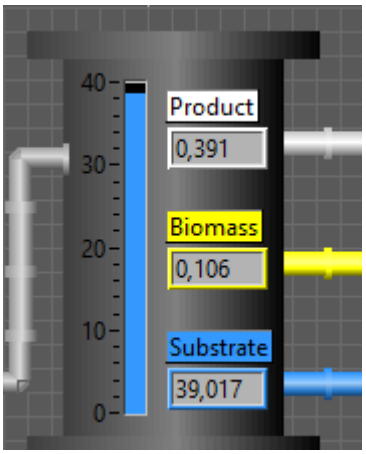
W tym podpanelu możliwe jest również ustawienie warunków początkowych opisywanych przy okazji przełącznika *Initial conditions*. Dodatkowo możliwe jest również ustawienie parametru Dt (ang. *deltaa time*), który opisuje rozdzielczość czasową symulacji oraz parametru NIS wykorzystywanego w metodzie UDYN opisanej w sekcji 3.2.2.

Za pomocą podpanelu zakresów symulacji możliwe jest ustalenie czasu symulacji pojedynczego wymuszenia (dla stałych D i S_{in}).

Czas kształtowania odpowiedzi (ang. *Response formation time*) określa okres czasu symulacji, po upływie którego zakładamy, że obiekt osiągnął stan ustalony (wartość stała lub oscylacje o niegasnącej amplitudzie). Czas pomiaru (ang. *Measurement time*) oznacza okres czasu, podczas którego wyznaczane są wartości: maksymalna, minimalna oraz średnia w stanie ustalonym obiektu. Łączny czas symulacji to suma czasu kształtowania odpowiedzi i czasu pomiaru.



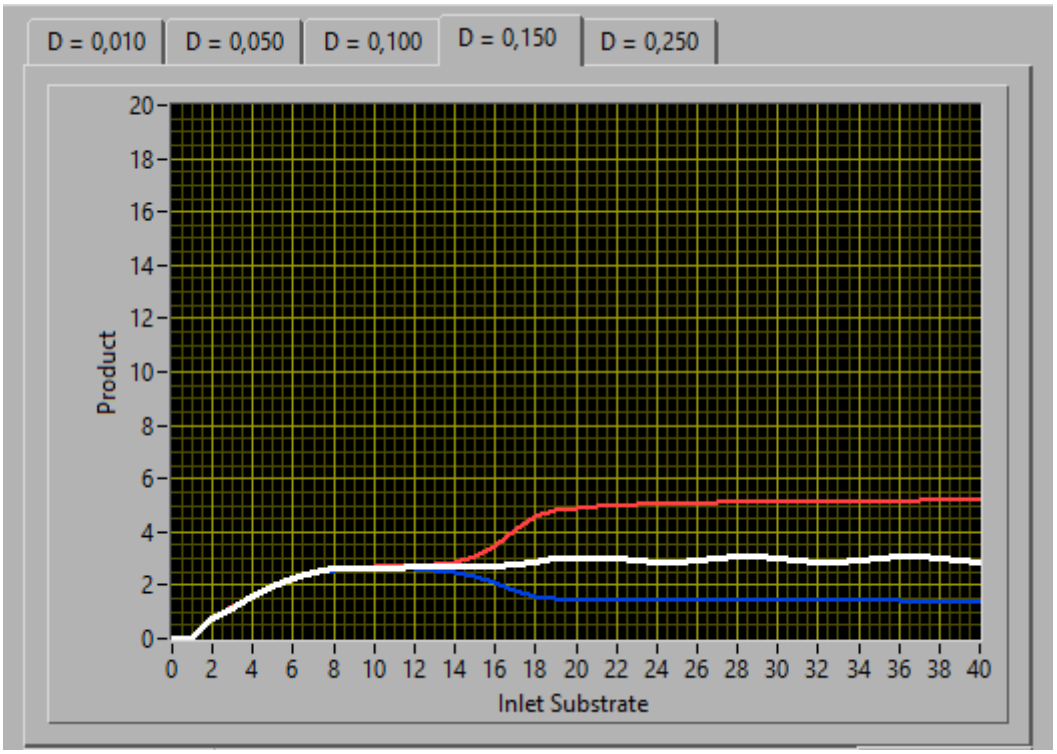
Rysunek 5: Podpanel zakresów symulacji



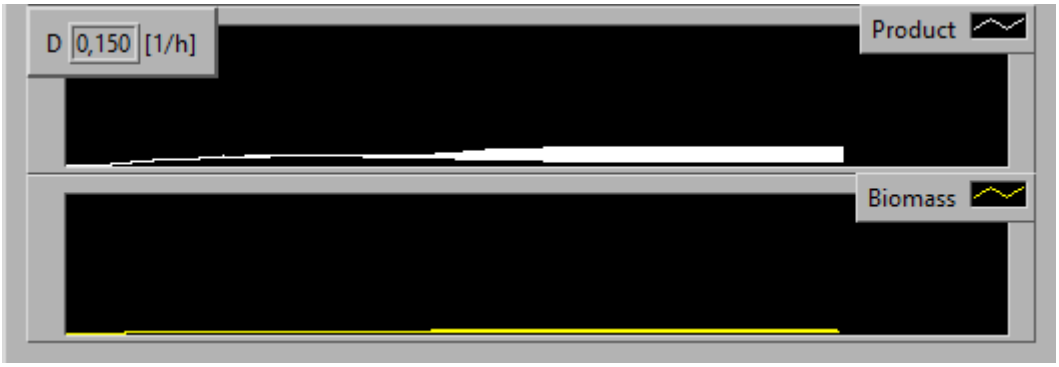
Rysunek 6: Wizualizacja obiektu reaktora

Ostatnim elementem sekcji ustawień panelu przedniego programu jest wizualizacja rozpatrywanego obiektu bioreaktora (Rysunek 6). Jest to schematyczne przedstawienie obiektu w postaci czarnej skrzynki z 1 wejściem - substratem wejściowym (kolor szary) oraz 3 wyjściami - produktem (kolor biały), biomasą (kolor żółty) oraz nieprzetworzonym substratem (kolor niebieski). Dodatkowo, suwak po lewej stronie rysunku reaktora pokazuje zależności pomiędzy stężeniami poszczególnych substancji. Szerokość poszczególnych pasków odpowiada ich stę-

żeniom, a kolory oznaczają konkretną substancję.

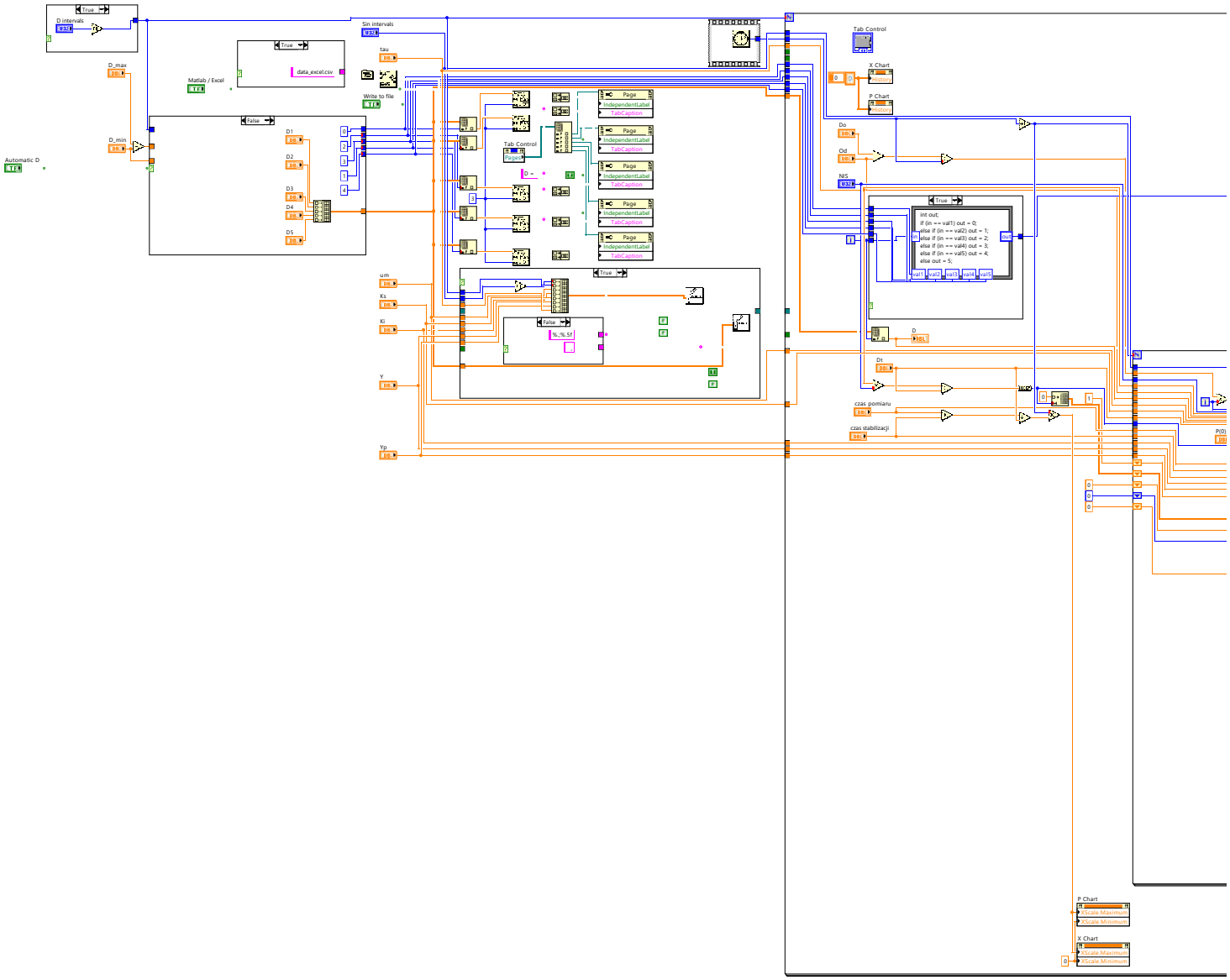


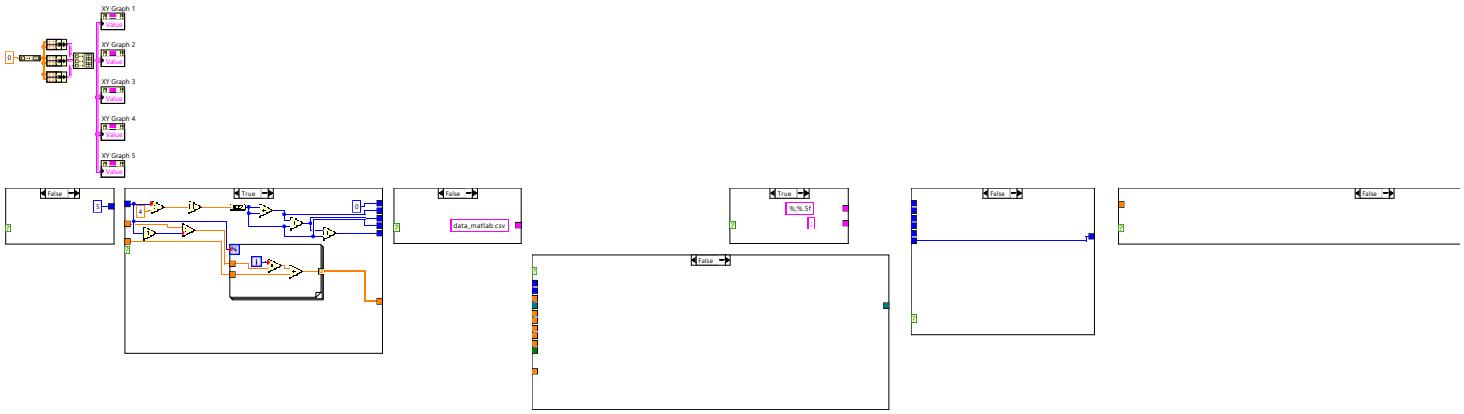
Rysunek 7: Wykresy charakterystyk roboczych w poszczególnych zakładkach

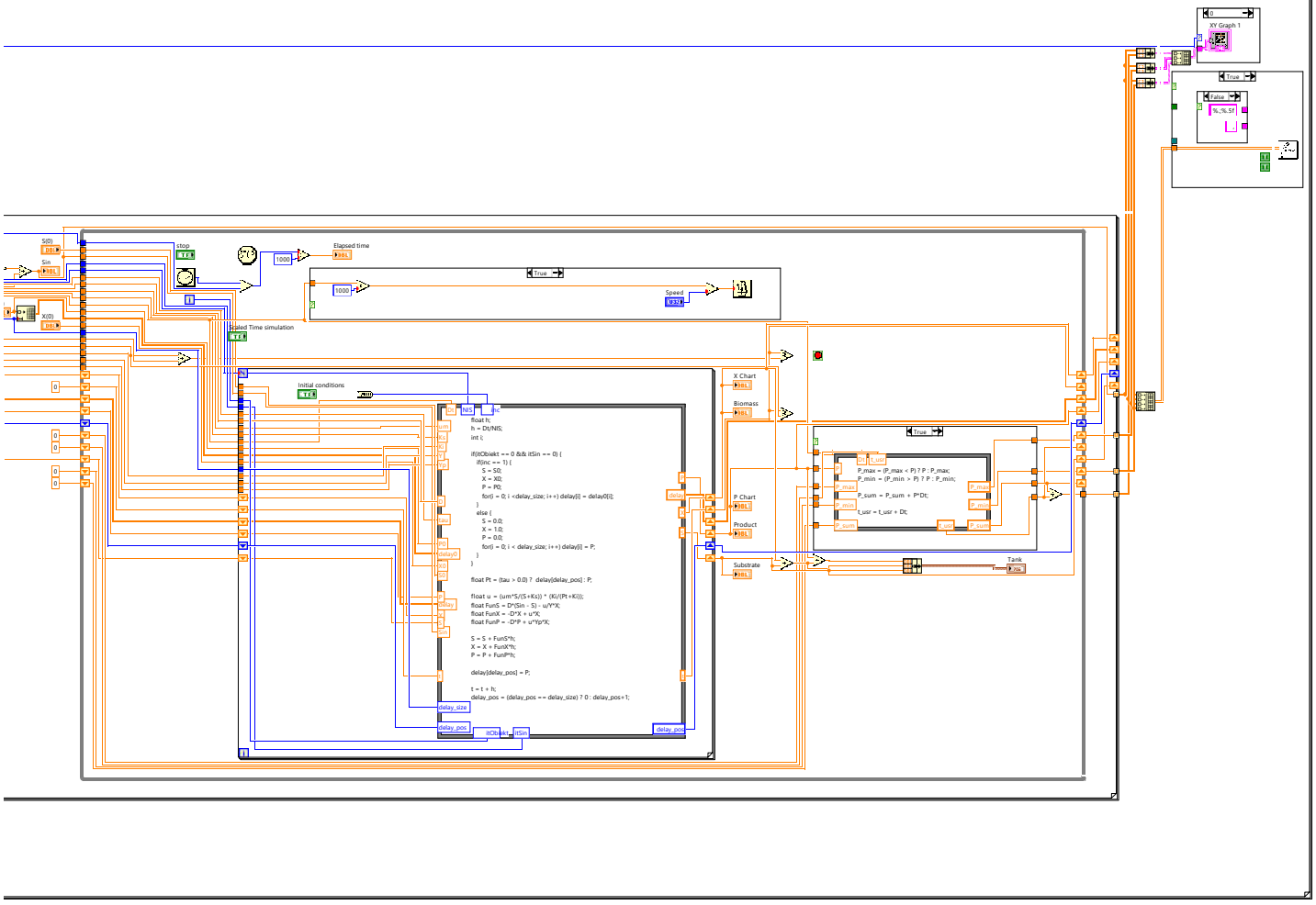


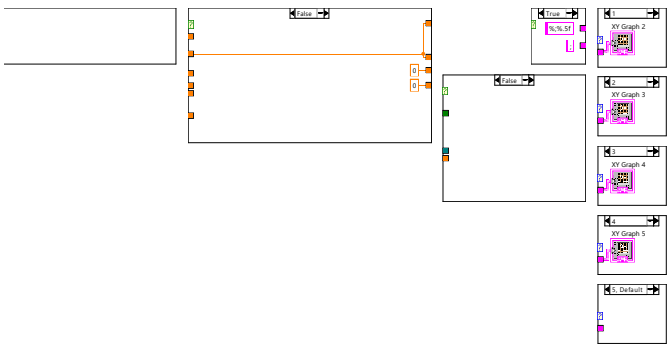
Rysunek 8: Wykresy przebiegu symulacji dla wybranego zakresu S_{in} i stałego D

Sekcja przebiegów panelu przedniego pozwala na podgląd danych na wykresach. Dwa dolne wykresy przedstawiają historię produktu oraz biomasy dla zmieniającego się substratu wejściowego S_{in}. Są generowane osobno dla każdej wartości parametru szybkości rozcieńczania D, przy czym aktualna jest wyświetlana w rogu wykresu. Zestaw 5 górnych charakterystyk roboczych jest rozdzielony w zakładkach o tytułach odpowiadających odpowiadającym im wartościom D. Na tych wykresach wartość średnia oscylacji dla danego S_{in} jest przedstawiona za pomocą koloru białego, wartość maksymalna - czerwonego, a wartość minimalna - granatowego.


D intervals
Sin intervals
tau
D_max
D_min
Automatic D
Matlab / Excel
Write to file
data_excel.csv
Tab Control
X Chart
P Chart
Do
Od
NIS
D1
D2
D3
D4
D5
um
Ks
Ki
Y
Yp
Di
D
czas pomiaru
czas stabilizacji
int out:
if (in == val1) out = 0;
else if (in == val2) out = 1;
else if (in == val3) out = 2;
else if (in == val4) out = 3;
else if (in == val5) out = 4;
else out = 5;
%.%.5f
P(0)


XY Graph 1
XY Graph 2
XY Graph 3
XY Graph 4
XY Graph 5
data_matlab.csv
%.%.5f

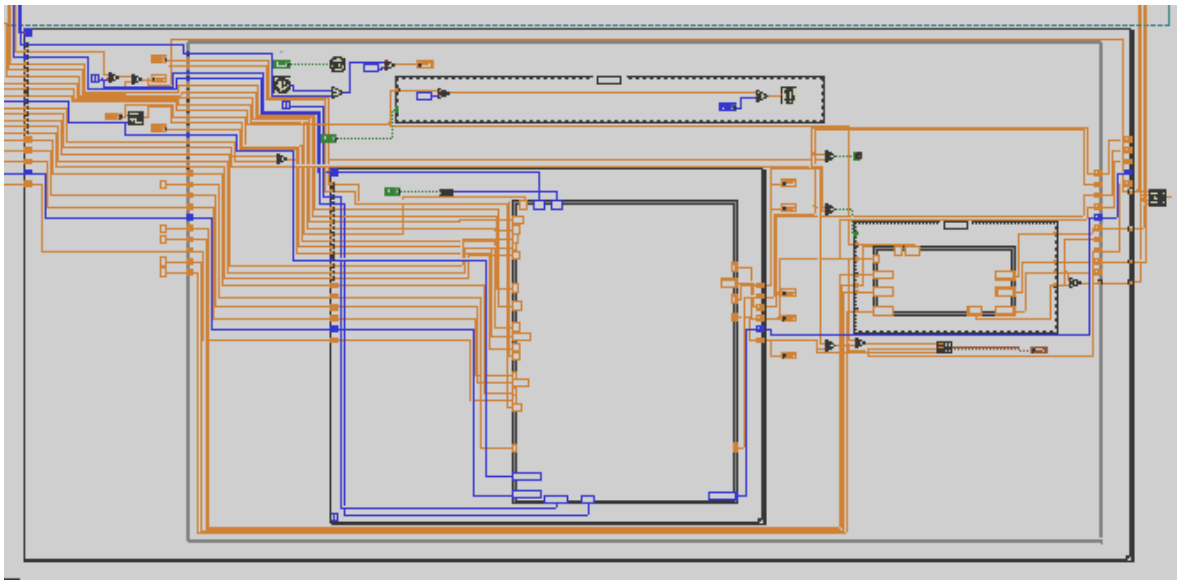

S(0)
Sin
X(0)
stop
Elapsed time
1000
True
Speed
Speed Time simulation
Initial conditions
X Chart
Biomass
P Chart
Product
Substrate
Tank
XY Graph 1
False
%.%.5f
float h;
h = Dt/NIS;
int i;
if(itObiekt == 0 && itSin == 0) {
if(inc == 1) {
S = S0;
X = X0;
P = P0;
for(i = 0; i <delay_size; i++) delay[i] = delay0[i];
}
else {
S = 0.0;
X = 1.0;
P = 0.0;
for(i = 0; i < delay_size; i++) delay[i] = P;
}
}
float Pt = (tau > 0.0) ? delay[delay_pos] : P;
float u = (um*S/(S+Ks)) * (Ki/(Pt+Ki));
float FunS = D*(Sin - S) - u/Y*X;
float FunX = -D*X + u*X;
float FunP = -D*P + u*Yp*X;
S = S + FunS*h;
X = X + FunX*h;
P = P + FunP*h;
delay[delay_pos] = P;
t = t + h;
delay_pos = (delay_pos == delay_size) ? 0 : delay_pos+1;
P_max = (P_max < P) ? P : P_max;
P_min = (P_min > P) ? P : P_min;
P_sum = P_sum + P*Dt;
t_usr = t_usr + Dt;


False
True
%.%.5f
XY Graph 2
XY Graph 3
XY Graph 4
XY Graph 5
5. Default

3.2 Diagram blokowy

Diagram blokowy jest graficznym przedstawieniem kodu programu symulatora i odpowiada za cały proces symulacji, a także komunikację z interfejsem użytkownika na panelu przednim. Sam kod składa się z 4 zagnieżdżonych podstawowych pętli: pętli realizującej metodę UDYN, głównej pętli obiektu oraz dwóch pętli odpowiadających za zmianę S_{in} i D.

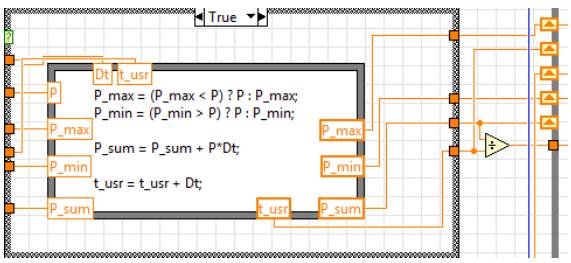
3.2.1 Pętla główna obiektu



Rysunek 9: Obszar głównej pętli

Pętla główna programu jest pętlą o niezdefiniowanej liczbie iteracji, która jest zatrzymywana po osiągnięciu zakładanego czasu symulacji (suma czasów kształtowania odpowiedzi i pomiaru). Jej głównym elementem jest główny blok kodu (Rysunek 11), który odpowiada za rozwiązywanie numeryczne równań stanu nieustalonego.

Dodatkowo występuje blok kodu wyznaczający wartość maksymalną, minimalną i średnią produktu w procesie, który uaktywnia się po upływie czasu kształtowania odpowiedzi i działa przez czas pomiaru. Dodatkowym blokiem jest układ skalowanego czasu opisany w sekcji 3.1. Wykorzystuje on bloczek *Wait Until Next ms Multiple*, który uruchamia program pętli co najmniejszą

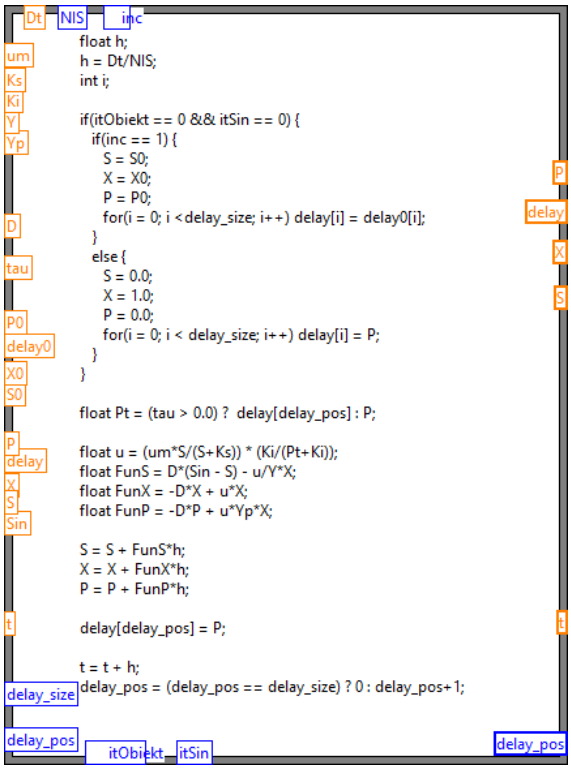


Rysunek 10: Kod wyznaczający wartość maksymalną, minimalną oraz średnią

możliwą wielokrotność zadanego mnożnika w [ms]. Mnożnik jest wyliczany na podstawie parametru użytkownika *Speed* oraz rozdzielczości symulacji Dt, co sprawia, że jest od niej niezależny - zmiana Dt nie powoduje spowolnienia/przyspieszenia symulacji.

3.2.2 Metoda UDYN

Całkowanie numeryczne jest procesem łatwo podatnym na błędy - małe odchyłki sumują się skutkując rosnącymi błędami wraz z upływem czasu. W niektórych przypadkach może to skutkować nawet błędnym charakterem odpowiedzi obiektu. W celu niedopuszczenia do takiej sytuacji stosuje się zmniejszony krok symulacji Dt (zwiększoną rozdzielczość czasową), co jednak powoduje zwiększenie ilości przetwarzanych danych.



Rysunek 11: Kod bloku głównego programu

Rozwiązaniem tego problemu jest metodyka UDYN (ang. *Unify DYNamics*) [1]. Zakłada ona osobny krok obserwacji i osobny krok całkowania. Krok obserwacji Dt opisuje czas aktualizacji danych z punktu widzenia z zewnątrz (np. dla pętli nadrzędnych, dla wyznaczania minimum i maksimum), natomiast mniejszy krok całkowania h opisuje różnicę czasu pomiędzy kolejnymi iteracjami w równaniu (9). Zależność pomiędzy tymi wielkościami jest opisana przez liczbę całkowitą NIS:

$$h = \frac{Dt}{NIS} \tag{14}$$

W programie metoda ta została zrealizowana poprzez dodatkową pętlę *for* wykonującą się NIS razy. Dodatkowo, wewnątrz kodu bloku głównego parametr h używany do dalszych obliczeń jest wyznaczany dynamicznie, co sprawia, że parametr NIS może być zmieniany przez użytkownika w trakcie trwania symulacji w zależności od potrzeb.

3.2.3 Realizacja opóźnienia

Opóźnienie jest członem układu, którego nie da się w prosty sposób zdyskretyzować na potrzeby całkowania numerycznego. Dlatego też wymagane jest inne podejście, a jednym z najczęstszych jest aproksymacja za pomocą n zmiennych pomocniczych [6]. Wtedy problem ten sprowadza się do rozwiązania układu n liniowych równań różniczkowych. W projekcie wykorzystano metodę linii *CTDS* (ang. *Continous Time - Discrete Space*) wraz ze schematem *2U* (ang. *2-point upwind formula*):

$$\frac{dP_\tau}{dt} = -\frac{1}{\tau}\frac{\partial P_\tau}{\partial z} \tag{15}$$

$$\frac{\partial P_\tau}{\partial x} = \frac{x_j - x_{j-1}}{\Delta z}; \qquad j = 1, ..., n \tag{16}$$

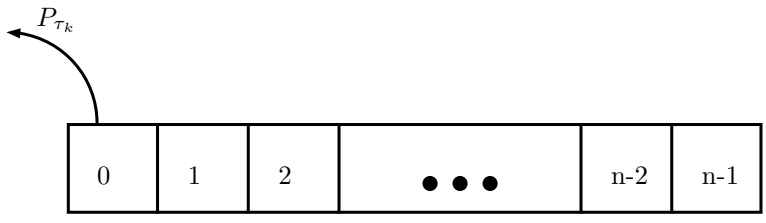
$$\frac{dp_j}{dt} = \frac{n}{\tau}(p_{j-1} - p_j); \qquad j = 1, ..., n \tag{17}$$

Przedstawione rozwiązanie umożliwiało uzyskanie poprawnych wyników zarówno przy $n = 100$, jak i $n = 20$. Niestety, konieczność wyznaczania n równań przy każdej iteracji pętli sprawiała, że algorytm działał bardzo wolno, co było szczególnie odczuwalne przy wykreślaniu charakterystyk dla dużej liczby przedziałów D i S_{in}.

Aby zmniejszyć złożoność czasową zdecydowano się skorzystać z karuzeli Franksa opisanej w [7]. Rozwiązanie to wykorzystuje większe zasoby pamięci w celu znacznego uproszczenia obliczeń i przyspieszenia działania programu.

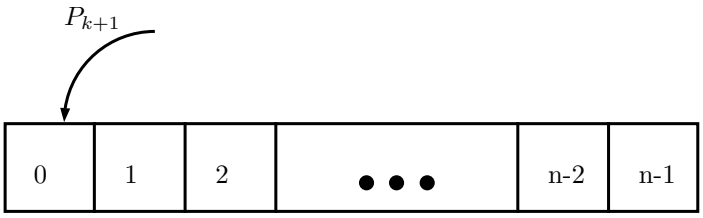
Karuzela Franksa wykorzystuje tablicę przechowującą poprzednio wyznaczone wartości produktu P, aby w razie potrzeby pobrać P_τ jako P z odpowiednio odległej komórki. Dlatego minimalny rozmiar tablicy definiowany jest jako $n = \frac{\tau}{Dt} * NIS$.

Zakładając wstępnie wypełnioną tablicę o rozmiarze n, numerowaną od 0 i aktualną iterację pętli symulacji k równą wielokrotności n (aktualna komórka równa 0), algorytm karuzeli Franksa zaimplementowany w programie działa następująco:



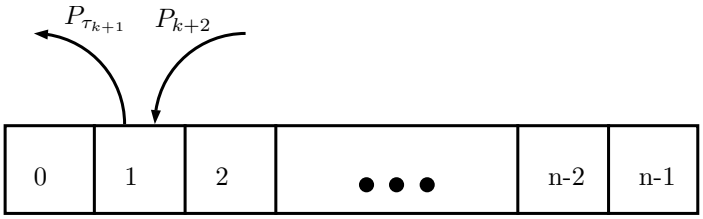
Pobranie danych z komórki 0

1. Następuje pobranie wartości P_τ z aktualnie wybranej komórki,



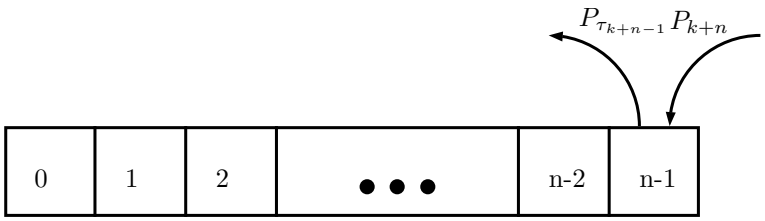
Zapis danych do komórki 0

2. Obliczenie nowej wartości P i jej zapis do aktualnej komórki,



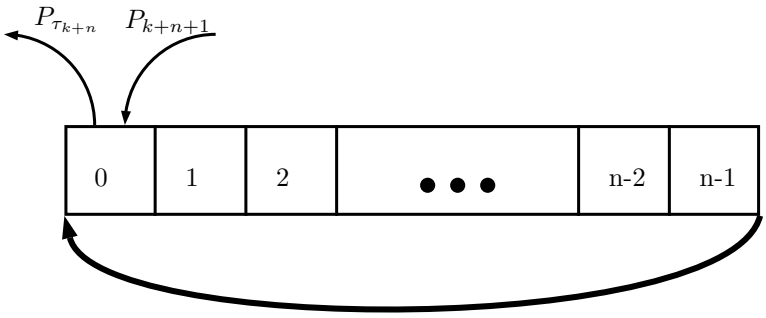
Pobranie i zapis danych do komórki 1

3. Wybranie kolejnej komórki, pobranie wartości P_τ i zapis wartości P,



Pobranie i zapis danych do komórki n-1

4. Powtórzenie punktu 3. dla kolejnych komórek w tablicy



Pobranie i zapis danych do komórki 0

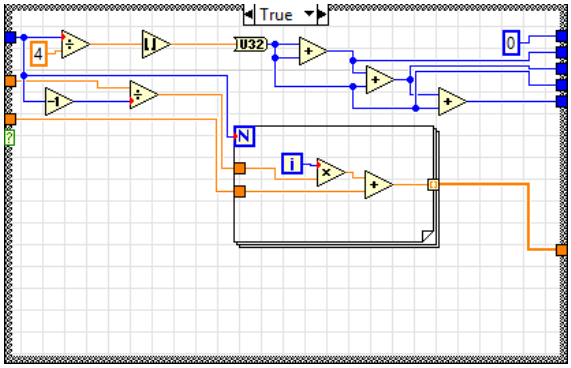
5. Po zapisie danych do ostatniej komórki n-1, wybierana jest znowu komórka 0 i cały proces jest powtarzany

Jak łatwo zauważyć, $P_{\tau_{k+n}}$ jest równe P_{k+1}, czyli zapisaną wcześniej do komórki wartość wyciągamy z niej ponownie dokładnie po n iteracjach, a więc τ jednostkach czasu.

Zysk czasowy jest znaczny, ponieważ 20 powtórzeń obliczania inercji w każdej iteracji pętli dla schematu *2U* zastępowane jest tylko 3 operacjami - odczytu, zapisu i inkrementacji wskaźnika aktualnej komórki.

3.2.4 Określanie charakterystyk roboczych

Mając wyznaczoną maksymalną, minimalną i średnią wartość produktu P dla danych parametrów należy je zmienić i wykonać kolejną symulację. Realizowane jest to poprzez 2 pętle *for* otaczające pętlę główną. Jedna zmienia parametr D w zadanym przedziale, a druga parametr S_in.



Rysunek 12: Blok generacji tablicy wartości D i wyboru przebiegów do wyświetlenia

Potrzeba obsługi zarówno wartości D wpisanych ręcznie, jak i wygenerowanych automatycznie wymusiła wykorzystanie tablicy, która przechowuje wszystkie rozpatrywane wartości. W przypadku wartości określanych ręcznie tablica przyjmuje rozmiar 5 i bezpośrednio wpisywane są do niej wartości podane przez użytkownika. Gdy została wybrana generacja automatyczna wyznaczany jest krok w postaci:

$$krok = \frac{D_{max} - D_{min}}{\text{l. przedziałów}}, \tag{18}$$

który następnie jest iteracyjnie sumowany z D_{min}, aby utworzyć pełen przedział wartości od D_{min} do D_{max} podzielony na odpowiednią liczbę przedziałów. Dodatkowo, na podobnej zasadzie wybierane są indeksy charakterystyk do wyświetlenia: D_{min}, 25%, 50% i 75% zakresu oraz D_{max}.

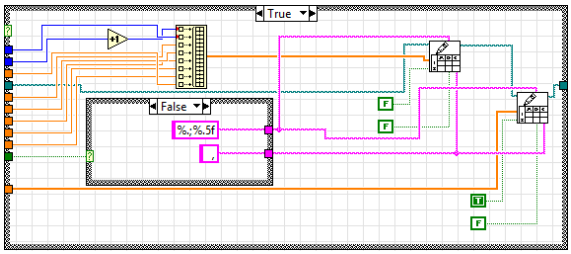
Wartości S_{in} generowane są na analogicznej zasadzie. Jednakże, zamiast do tablicy, podawane są bezpośrednio do głównej pętli symulatora.

3.2.5 Zapis do pliku

Zapis do pliku odbywa się w dwóch częściach: zapis parametrów procesu i zapis danych wyjściowych. Pierwsza część umieszcza w pliku wynikowym (*data_matlab.csv* lub *data_Excel.csv*)

w pierwszym wierszu liczbę różnych wartości S_{in} i liczbę różnych wartości D, a także parametry stałe: μ_m, K_S, K_i, Y, Y_P oraz opóźnienie τ. Druga linia zawiera kolejne wartości D, dla których była przeprowadzona symulacja.

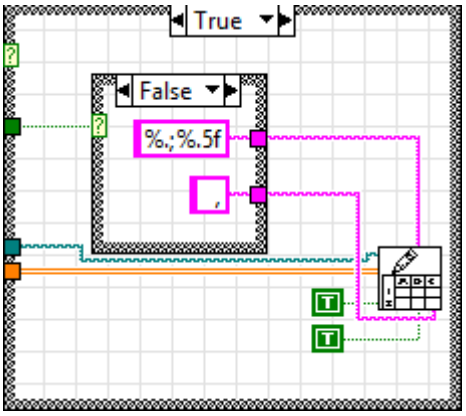
Kolejna część zapisuje połączone tablice z wynikami symulacji dla całego przedziału S_{in} i kolejnych D. W kolejnych kolumnach znajdują się S_{in}, P_r, P_{max} i P_{min}.



Rysunek 13: Zapis do pliku parametrów procesu

Niezależnie od wybranego ustawienia *Matlab/Excel* plik wynikowy jest zapisany w formacie CSV (ang. *Comma-separated value*). Niestety, standard zapisu danych w pliku różni się w zależności od implementacji:

- MATLAB wymaga, aby separatorem dziesiętnym była kropka, a poszczególne liczby były oddzielone przecinkami,
- Microsoft Excel akceptuje pliki z liczbami rozdzielonymi średnikami, a operator dziesiętny zależy od ustawień systemowych



Rysunek 14: Zapis do pliku wyników symulacji

Z tego powodu program oferuje wybór preferowanego sposobu zapisu pliku. Należy jednak zauważyć, że dołączony skrypt do generacji wykresów 3D - *CFPlplot.m*, nie będzie współpracował wyłącznie z plikiem sformatowanym dla programu MATLAB.

4 Przykładowe wyniki symulacji

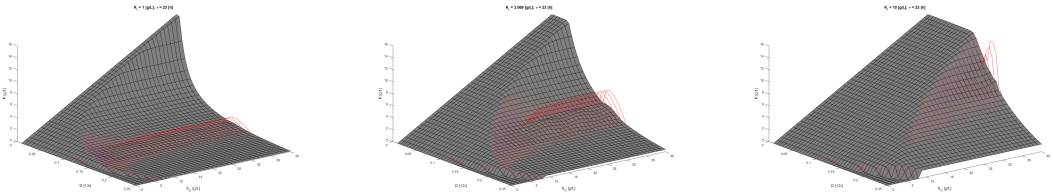
Po zakończeniu symulacji i zapisie wygenerowanych danych do pliku sformatowanego dla programu MATLAB, istnieje możliwość użycia załączonego skryptu *CFPlplot.m* w celu generacji wykresów 3D. Skrypt przyjmuje jeden parametr - ścieżkę do odpowiedniego pliku CSV. W przypadku niepodania parametru, skrypt przyjmuje domyślną wartość - plik *data_matlab.csv* w katalogu skryptu.

Przy wykreślaniu przykładowych przebiegów skorzystano z parametrów procesu wykorzystanych w [2]. W celu lepszego ukazania obszarów oscylacji użyto mniejszego opóźnienia $\tau = 23[h]$.

Tabela 1: Przykładowe wartości parametrów procesu

μ_m	K_S	K_i	Y	Y_P	τ
0.282	0.025	3.069	0.108	3.68	23

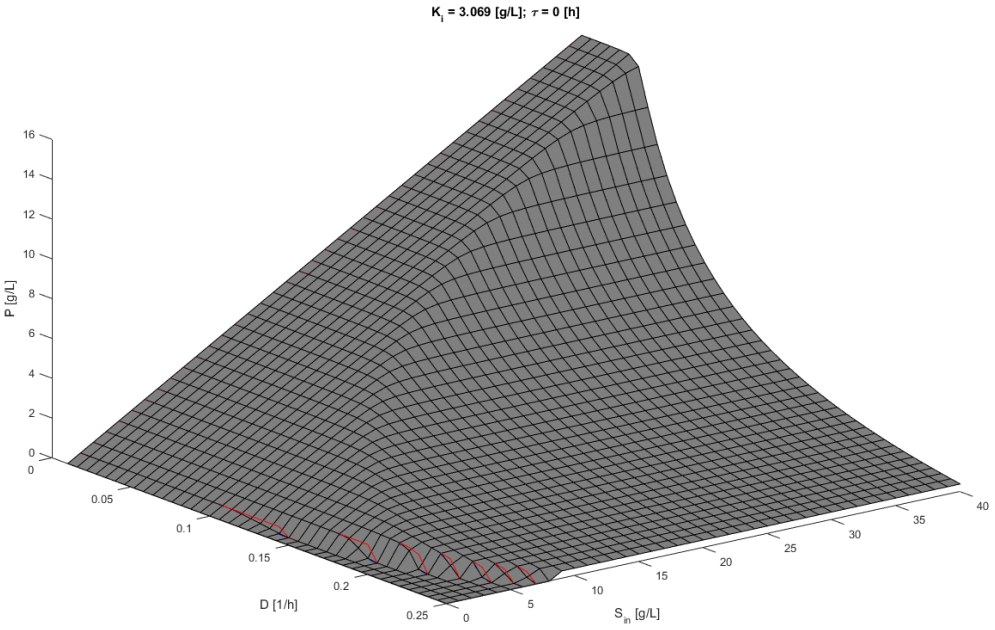
Wykorzystując te dane można porównać wpływ stałej hamowania K_i:



$K_i = 1$ $K_i = 3,069$ $K_i = 10$

Rysunek 15: Wpływ K_i na charakterystyki robocze obiektu

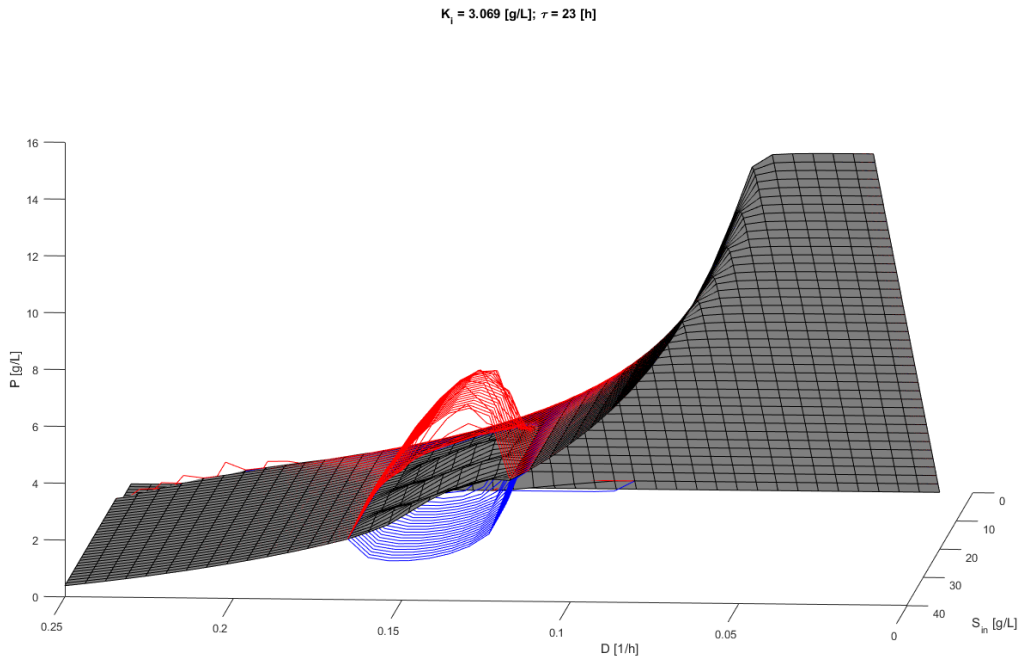
Można również wykreślić przebieg charakterystyki bez opóźnienia:



Rysunek 16: Przebieg charakterystyki przy braku opóźnienia

Wykorzystując możliwości środowiska MATLAB istnieje również możliwość analizy mak-

simów i minimów oscylacji:



Rysunek 17: Charakterystyka obrócona, aby pokazać maksima i minima

5 Podsumowanie

Proces fermentacji z opóźnionym hamowaniem produktem jest wysoce nieliniowym procesem, dla którego nie jest możliwe analityczne wyznaczenie charakterystyk w stanie ustalonym. W tym celu stosuje się wielokrotną symulację poprzez rozwiązywanie numeryczne modelu.

Zaproponowany program jest w stanie symulować rozpatrywany proces dla zadanych parametrów oraz wyznaczać charakterystyki robocze i przedstawiać je na wykresach. Poprzez zastosowanie karuzeli Franksa wyniki otrzymywane są w bardzo krótkim czasie. Intuicyjny interfejs użytkownika sprzyja szybkiej analizie procesu, a mnogość opcji eksportu danych pozwala kontynuować analizę za pomocą specjalistycznego oprogramowania.

Bibliografia

[1] M. Metzger, *Modelling, simulation and control of continuous processes*, Edition of Jacek Skalmierski, Gliwice, 2000.

[2] P. Skupin, M. Metzger, *Oscillatory behavior control in continous fermentation processes*, IFAC-PapersOnLine 48-8, pp. 1114–1119, 2015.

[3] F.W. Bai, X.Q. Zhao, *High Gravity Ethanol Fermentations and Yeast Tolerance*, Microbial Stress Tolerance for Biofuels. Microbiology Monographs, 22, pp. 117–135, 2012.

[4] P. Skupin, *Simulation approach for detection of the self-sustained oscillations in continuous culture*, Proceedings of the 11th WSEAS International Conference on Mathematics and Computers in Biology and Chemistry (MCBC) Iasi, Romania, pp. 80–85, 2010.

[5] R. Luedeking, E.L. Piret, *A kinetic study of the lactic fermentation. Batch process at controlled pH*, Biotechnology and Engineering, 1 (4), pp. 393–412, 1959.

[6] W.T. Mocek, R. Rudnicki, E.O. Voit, *Approximation of delays in biochemical systems*. Mathematical Biosciences, 198 (2), pp. 190–216, 2005.

[7] R.G.E. Franks, *Modeling and simulation in chemical engineering*, Wiley-Interscience, London, 1972.

Skrypt *CFPlplot.m*

```
function CFPlplot(path)
    if nargin < 1
        path = 'data_matlab.csv';
    end

    [Sin, D, data, param] = csv2matlab(path);

    figure;
    for i = 1:size(Sin)
        temp = ones(size(D))*Sin(i);
        plot3(temp, D, transpose(data(i,:,3)), 'b');
        hold on
        plot3(temp, D, transpose(data(i,:,2)), 'r');
        hold on
    end

    surf(Sin, D, transpose(data(:,:,1)));
    colormap([0.5 0.5 0.5]);
    axis ij;
    xlabel('S_{in} [g/L]');
    ylabel('D [1/h]');
    zlabel('P [g/L]');
    title(['K_i = ' num2str(param(3)) ' [g/L]; \tau = ' num2str(
       param(6)) ' [h]']);
end

function [Sin, D, data, param] = csv2matlab(path)
    temp = csvread(path, 0, 0, [0 0 0 7]);
    paramSin = temp(1); paramD = temp(2); param = temp(3:8);

    D = transpose(csvread(path, 1, 0, [1 0 1 paramD-1]));
    Sin = csvread(path, 2, 0, [2 0 paramSin+1 0]);
```

```
    data = zeros(paramSin, paramD, 3);

    for i = 1:3
        for d = 1:paramD
            data(:, d, i) = csvread(path, 2+paramSin*(d-1), i, [2+
                paramSin*(d-1) i 1+paramSin*d i]);
        end
    end
end
```

Spis rysunków

1 Panel przedni programu *CFPI.vi* 8
2 Podpanel ustawień trybu pracy symulatora 8
3 Podpanel ręcznych wartości szybkości rozcieńczania 9
4 Podpanel ustawień parametrów procesu 10
5 Podpanel zakresów symulacji 10
6 Wizualizacja obiektu reaktora 10
7 Wykresy charakterystyk roboczych w poszczególnych zakładkach 11
8 Wykresy przebiegu symulacji dla wybranego zakresu S_{in} i stałego D 11
9 Obszar głównej pętli 14
10 Kod wyznaczający wartość maksymalną, minimalną oraz średnią 14
11 Kod bloku głównego programu 15
12 Blok generacji tablicy wartości D i wyboru przebiegów do wyświetlenia 18
13 Zapis do pliku parametrów procesu 19
14 Zapis do pliku wyników symulacji 19
15 Wpływ K_i na charakterystyki robocze obiektu 20
16 Przebieg charakterystyki przy braku opóźnienia 20
17 Charakterystyka obrócona, aby pokazać maksima i minima 21