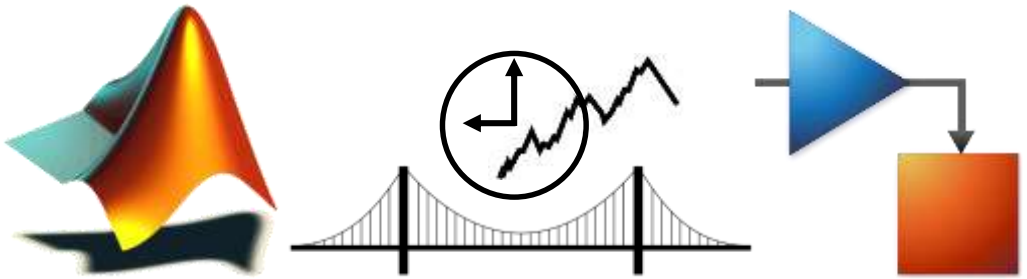
Time Series Forecasting on Simulink®

MathWorks® Japan
Application Engineering Group

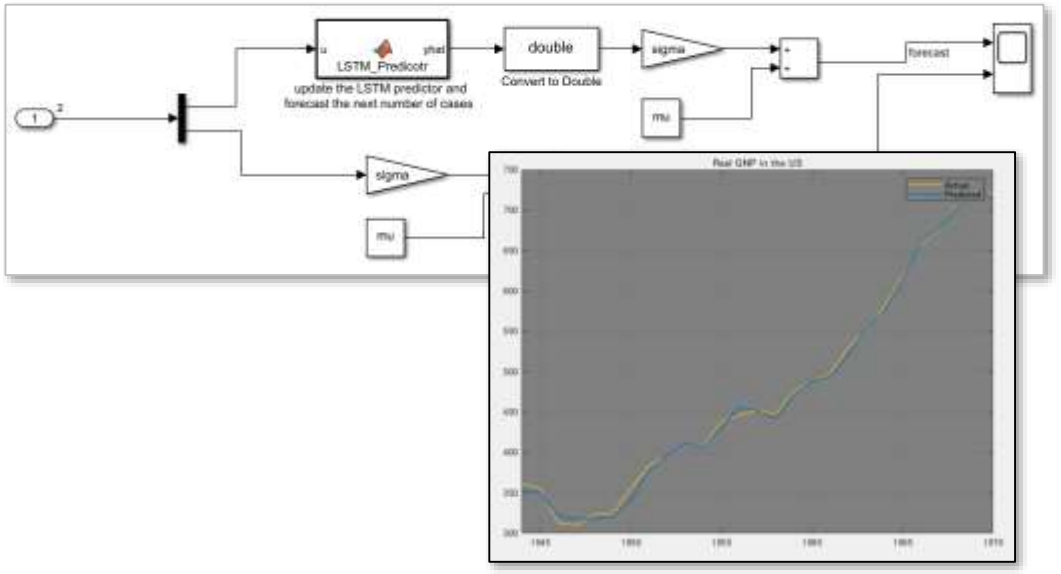
目的

- 時系列の予測を Simulink モデル上で！
  - Deep Learning を Simulink モデルに組み入れたい
  - 色々な時系列モデルを Simulink 上で試してみたい
  - MATLAB でサポートされる機能をモデルに実装したい

といった ご要望に対し「どのように実装するか」を具体例を交えながらご紹介します



Key Takeaways



- 時系列を再帰的に推定する機能を提供する
  - Deep Learning Toolbox™
  - Econometrics Toolbox™

に焦点を当てて MATLAB® Function ブロックを介して Simulink モデルに実装する方法を示します。しかし、これらの製品に限らず

  - Predictive Maintenance Toolbox™
  - Statistics and Machine Learning Toolbox™
  - System Identification Toolbox™

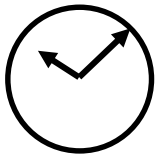
など任意の製品が提供する機能(特に回帰)にも応用することが可能です

Table of Contents

1. MATLAB と Simulink の違い

- MATLAB はアプリケーション開発言語という側面がありますが、一方の Simulink はブロック線図によって処理フローのロジックを構築するための環境を提供いたします。Simulink モデルは視覚的には理解しやすいですが MATLAB と異なり次のような特徴があります

  - シミュレーション環境の提供
    - モデル実行時には必ず時刻に関する微分方程式ソルバーが動作するため、常に「時刻」という概念が付いて回ります



  - モデルベース デザインを支える自動コード生成
    - Simulink Coder™ をはじめとするコード生成機能を提供する製品と連携が図られているため MATLAB に比べて扱えるデータ型が限定的
    - MATLAB はデータ型の異なる数値同士の演算を施しても自動的にキャストしてくれますが Simulink は明示的に変換する必要があります

2. MATLAB Function ブロック

MATLAB Function

- MATLAB Function ブロックは、カスタムのMATLAB プログラムを Simulink モデルに実装する機能を提供するだけではなくコード生成までを実現いたします
  - ただし、あくまで Simulink の特徴より MATLAB がサポートする機能全てが MATLAB Function ブロックにて使用できる訳ではありません
    - https://www.mathworks.com/help/slci/ref/matlab-function-block-support.html
    - https://www.mathworks.com/help/slci/ref/matlab-function-block-constraints.html
  - Deep Learning Toolbox や Econometrics Toolbox が提供する関数を このブロックを介して使用する場合は、特に以下のことに注意を払う必要があります
    - サポート対象外のデータ型の使用
    - コード生成未対応関数の呼び出し

2. MATLAB Function ブロック（cont.）

MATLAB Function

Q サポート対象外のデータを使う方法は？

- MATLAB および Toolbox が提供するクラスのインタンスであるオブジェクトはほぼサポート対象外です。たとえば、次のような変数もサポート対象外に該当します
  - Deep Learning Toolbox が提供する SeriesNetwork
  - Econometrics Toolbox が提要する各時系列クラス
  - 関数ハンドル

```
function y = fun(u)
  :
y = 0;
  :
y = myfcn(u);
  :
end
```

カスタム関数の出力変数の型はチェックできないので事前にデータ型を宣言

A ラッパー関数を用意しこの内部で利用する

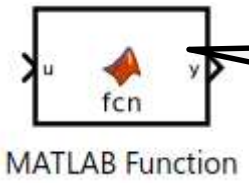
- この関数内では自由にデータ型を使用してよい

```
function yhat = myfcn(y_t)

net = evalin('base', 'net');
[newnet, yhat] = predictAndUpdateState(net, y_t);
assignin('base', 'net', newnet);
end
```

2. MATLAB Function ブロック（cont.）

Q コード生成未対応関数を呼び出すには？
- カスタム関数を含めコード生成未対応の関数は多数存在します

A 「coder.extrinsic」によってコード生成不要であることを宣言する事でビルド時のコード解析を回避




```
function yhat = LSTM_Predicotr(u)
 :
coder.extrinsic('myfcn');
yhat = myfcn(u);
 :
end
```

3. 実装前の検討

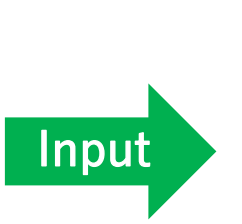
- 時系列モデルを Simulink モデルに実装する前に 2 つのテクニカルな課題を提起いたします

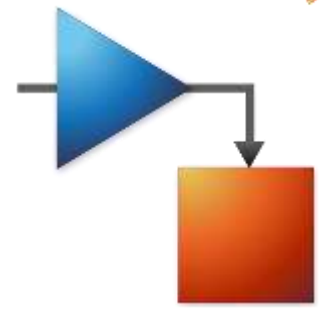
1. 入力となる時系列データをどのように Simulink モデルに渡すか
2. 学習済みモデルをどのように Simulink モデルに読み込ませるか

前者の時系列データをどのように Simulink モデルに渡すかは、どの時系列モデルを採用するにしても考慮しなければならない課題です。一方、後者の学習済みモデルに関しては、特に Deep Learning のように事前に学習器を作成するようなケースは必要な検討事項です

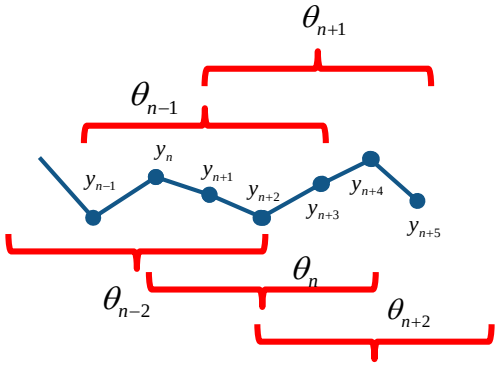
最後に、シミュレーションの際に必要なソルバー設定について言及いたします

時系列データと Simulink の挙動
部分時系列配列の用意





- 部分時系列
  - Window 幅 w = 5 で部分時系列へ分割する様子



	θ_1	θ_2	θ_3	⋯	θ_n	θ_{n+1}	⋯
1	y_1	y_2	y_3	⋯	y_n	y_{n+1}	⋯
2	y_2	y_3	y_4	⋯	y_{n+1}	y_{n+2}	⋯
3	y_3	y_4	y_5	⋯	y_{n+2}	y_{n+3}	⋯
:	:	:	:	⋱	:	:	:
w-1	y_{w-1}	y_w	y_{w+1}	⋯	y_{n+w-2}	y_{n+w-1}	⋯
w	y_w	y_{w+1}	y_{w+2}	⋯	y_{n+w-1}	y_{n+w}	⋯

    - ここでのシフト幅は「1」
  - Simulink モデルへの入力するための
    データ型として 構造体や Timetable 配列 などがありますが、ここでは2次元行列を
    ベース・ワークスペースに作成する方法を採用いたします

実装前の検討
部分時系列配列の用意 (cont.)

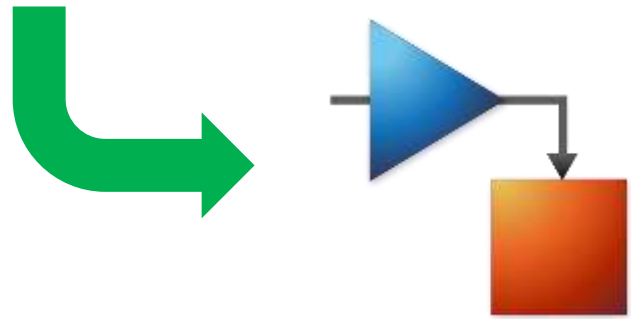
- Simulink における部分時系列の仕様
  - Simulink は 2 次元行列の場合、各タイムステップで「行単位」でインポート
    - → 前ページで用意した部分時系列を転置し左端に時刻の列を付与

	θ_1	θ_2	θ_3	$\cdots$	θ_n	θ_{n+1}	$\cdots$
1	y_1	y_2	y_3	$\cdots$	y_n	y_{n+1}	$\cdots$
2	y_2	y_3	y_4	$\cdots$	y_{n+1}	y_{n+2}	$\cdots$
3	y_3	y_4	y_5	$\cdots$	y_{n+2}	y_{n+3}	$\cdots$
:	:	:	:	$\ddots$	:	:	:
w-1	y_{w-1}	y_w	y_{w+1}	$\cdots$	y_{n+w-2}	y_{n+w-1}	$\cdots$
w	y_w	y_{w+1}	y_{w+2}	$\cdots$	y_{n+w-1}	y_{n+w}	$\cdots$

t

転置

t_1	y_1	y_2	y_3	$\cdots$	y_{w-1}	y_w	θ_1
t_2	y_2	y_3	y_4	$\cdots$	y_w	y_{w+1}	θ_2
t_3	y_3	y_4	y_5	$\cdots$	y_{w+1}	y_{w+2}	θ_3
:	:	:	:	:	:	:	



時系列データと Simulink の挙動

Simulink モデルへ部分時系列を入力

- ここでは次の 2 つの方法で Simulink モデルへ 2 次元行列に加工した部分時系列配列をインポートする方法を示します

1. コンフィギュレーション パラメーターからの設定
2. From Workspace ブロックの利用

$$\Theta \coloneqq \begin{array}{|c|c|c|c|c|c|c|} \hline t_1 & y_1 & y_2 & y_3 & \cdots & y_{w-1} & y_w \\ \hline t_2 & y_2 & y_3 & y_4 & \cdots & y_w & y_{w+1} \\ \hline t_3 & y_3 & y_4 & y_5 & \cdots & y_{w+1} & y_{w+2} \\ \hline \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \hline \end{array}$$

時系列データと Simulink の挙動

Simulink モデルへ部分時系列を入力: Configuration Parameter

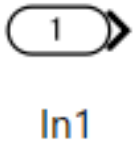
1. コンフィギュレーション パラメーターから設定する場合
   1. 「モデル化」タブ → 「モデル 設定」
   2. 「データのインポート/エクスポート」
   3. 「ワークスペースからの読み込み」にある「入力」をチェック
   4. 変数 t および u に関しては適宜変更
      - ただし、t に相当する変数はインポートする 2 次元行列の先頭列、u に相当する変数は 2 列目以降がそれぞれ対応

t_1	y_1	y_2	y_3	$\cdots$	y_{w-1}	y_w
t_2	y_2	y_3	y_4	$\cdots$	y_w	y_{w+1}
t_3	y_3	y_4	y_5	$\cdots$	y_{w+1}	y_{w+2}
:	:	:	:	:	:	:

t u

時系列データと Simulink の挙動

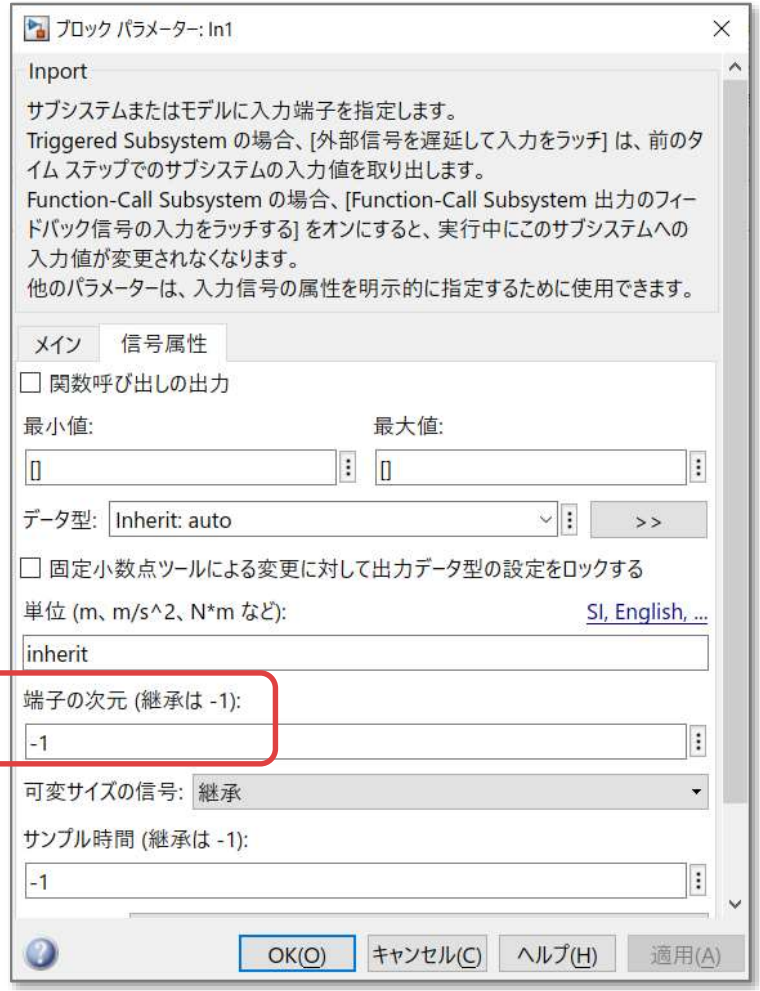
Simulink モデルへ部分時系列を入力: Configuration Parameter (cont.)



5. Inport ブロックを Simulink モデルへ配置
6. Inport ブロックの端子の次元を適切な値に編集
   - u の列数、つまり w に相当する値へ変更

t_1	y_1	y_2	y_3	$\cdots$	y_{w-1}	y_w
t_2	y_2	y_3	y_4	$\cdots$	y_w	y_{w+1}
t_3	y_3	y_4	y_5	$\cdots$	y_{w+1}	y_{w+2}
$:$	$:$	$:$	$:$	$:$	$:$	$:$

u に相当する列数 w



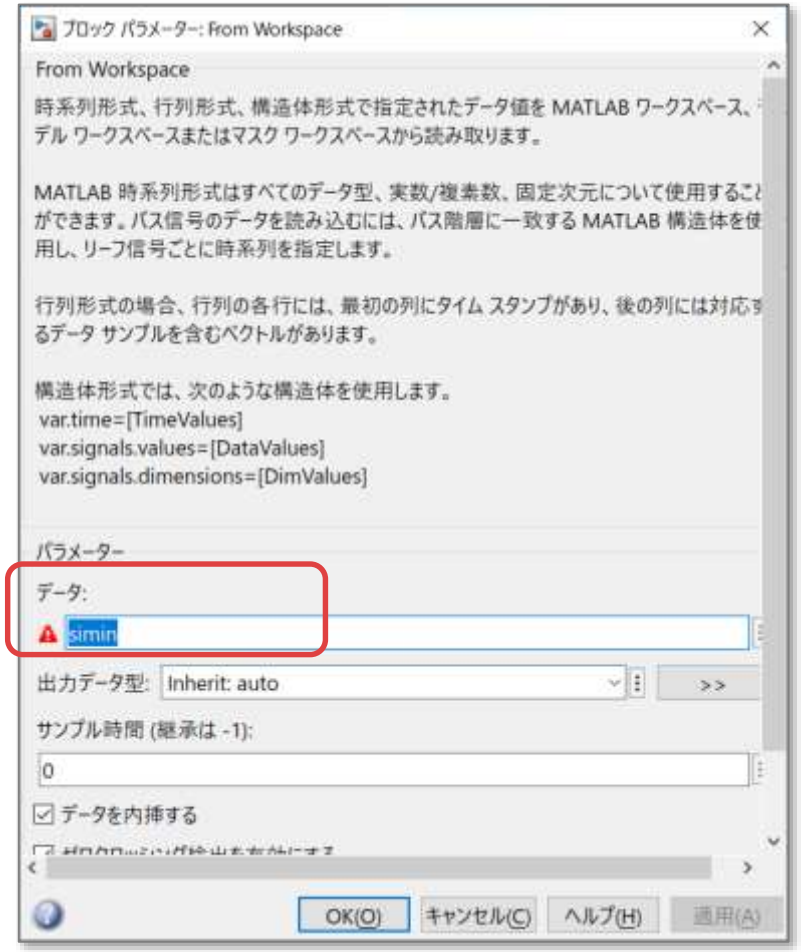
https://jp.mathworks.com/help/simulink/ug/load-data-to-root-level-input-ports-1.html#bsuwnz1

時系列データと Simulink の挙動

Simulink モデルへ部分時系列を入力: From Workspace ブロック

- From Workspace ブロックの場合
  1. From Workspace ブロックを Simulink モデルへ配置
  2. ベース・ワークスペースに作成した Θ に該当する
     ２次元行列の変数を「データ」に指定

$$\Theta = \begin{array}{|c|c|c|c|c|c|c|} \hline t_1 & y_1 & y_2 & y_3 & \cdots & y_{w-1} & y_w \\ \hline t_2 & y_2 & y_3 & y_4 & \cdots & y_w & y_{w+1} \\ \hline t_3 & y_3 & y_4 & y_5 & \cdots & y_{w+1} & y_{w+2} \\ \hline \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \hline \end{array}$$



事前学習済み学習器のインポートとエクスポート

- MathWorks 製品が提供する様々なクラスによって生成されるオブジェクトは、そのほとんどが Simulink モデルの上では扱えません。従って、学習済みオブジェクトのような場合は、MATLAB のベース ワークスペースを介して Simulink モデルにインポート/エクスポートするのが手軽です
  - 「2. MATLAB Function ブロック」で取り上げたように、この内部で直接扱えるクラスは限定的であるためラッパー関数を用意して間接的に呼び出します


```
function yhat = LSTM_Predicotr(u)

yhat = single(0);
coder.extrinsic('my_update_predict_fcn');
yhat = my_update_predict_fcn(u);

end
```

```
function yhat = my_udate_predict_fcn(y_t)

net = evalin('base', 'net');
[newnet, yhat] = predictAndUpdateState(net, y_t);
assignin('base', 'net', newnet);
end
```

ソルバーなど各種設定

t_1	y_1	y_2	y_3	$\cdots$	y_{w-1}	y_w
t_2	y_2	y_3	y_4	$\cdots$	y_w	y_{w+1}
t_3	y_3	y_4	y_5	$\cdots$	y_{w+1}	y_{w+2}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

- 右の表のような時刻情報を持つ 2 次元の部分時系列行列を Simulink に入力として渡しますので以下のようにモデルのパラメータを指定
  - ソルバーは固定ステップを選択しステップサイズは $\Delta t (:= t_{n+1} - t_n)$ を指定
  - 開始および終了時間をデータに合わせて適切な値を指定

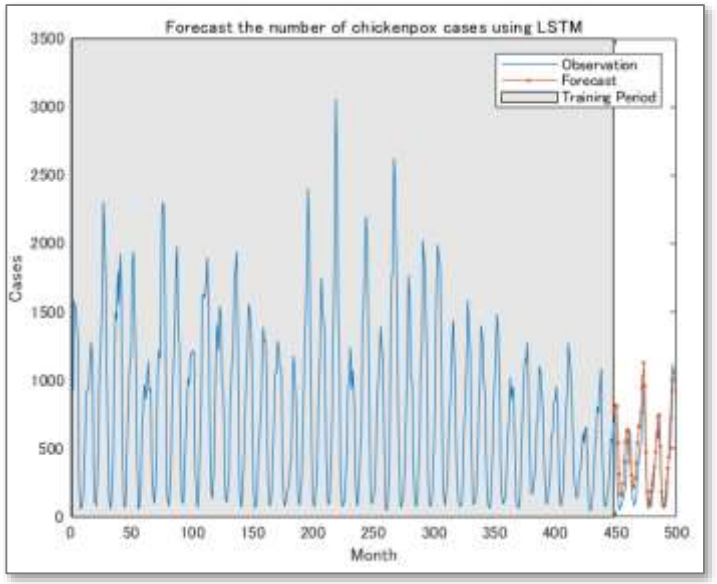
```
mdl = 'my_recursive_update_model';
set_param(mdl, 'Solver', 'FixedStepAuto');
set_param(mdl, 'FixedStep', '1');
set_param(mdl, 'StartTime', num2str(Time(1)));
set_param(mdl, 'StopTime',  num2str(Time(end)));
```

$\Delta t = 1$

Time(1) $:= t_1$

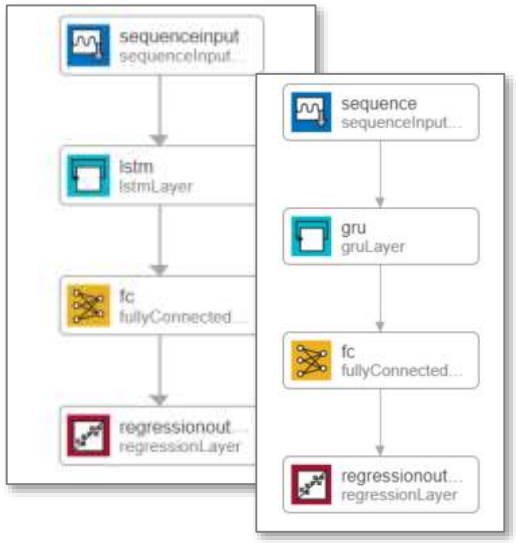
Time(end) $:= t_N$

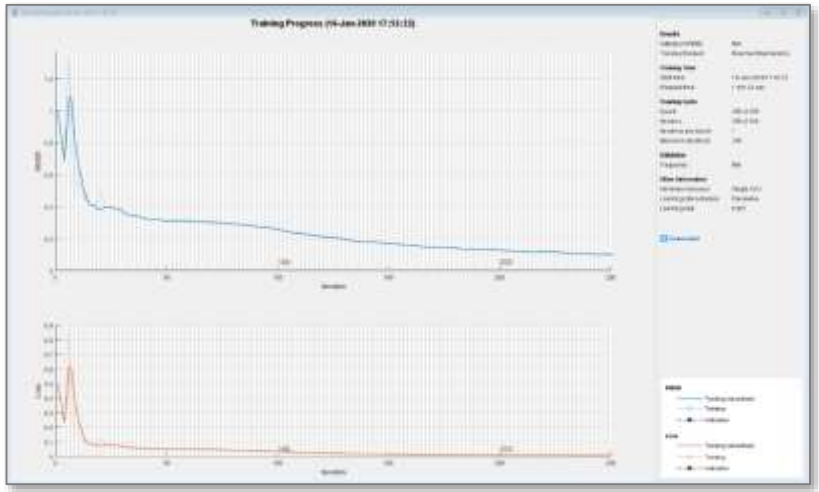
LSTM/GRU を実装する方法



1. サンプル「LSTM¥time_series_regression.mlx」、
   または「GRU¥time_series_regression.mlx」について
   - USにおける水痘症患者数の推移予測 (月次)
     - LSTM層またはGRU層を含んだネットワーク構築
     - 448ヶ月分のデータを使ったネットワークの学習
       - データに関しては正規化を実施
     - 最後の50ヶ月についてはネットワークを更新しながら翌月の患者数を予測
       参照) https://www.mathworks.com/help/deeplearning/ug/time-series-forecasting-using-deep-learning.html

2. 学習済みネットワークを準備
   1. Deep Network Designer などにより
      LSTM 層(or GRU層)を含む
      ネットワークを構築
   2. 事前学習済みネットワークを作成





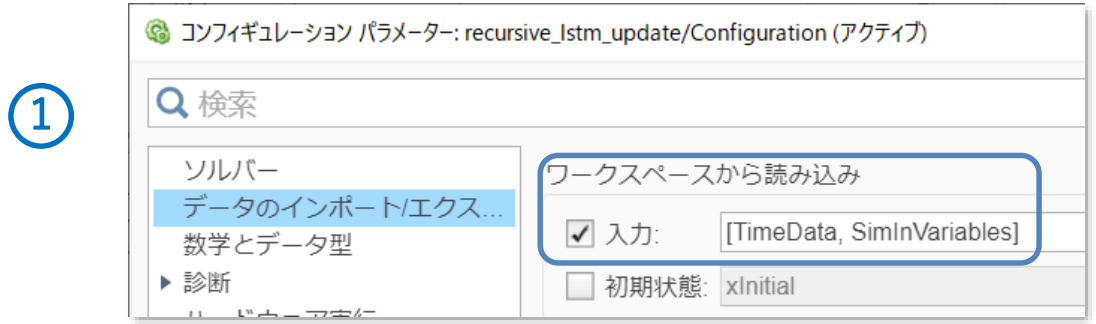
LSTM/GRU を実装する方法 (cont.)

3. Simulink モデルへ時系列配列を渡すための準備
   - 時刻データ配列 **`TimeData`** を作成
   - 部分時系列配列 **`SimInVariables`** を作成

TimeData **SimInVariables**

t_1	y_{t_1}	y_{t_2}
t_2	y_{t_2}	y_{t_3}
t_3	y_{t_3}	y_{t_4}
:	:	:

4. モデル作成のポイント
   - 「recursive_lstm_update.slx」を例にポイントを列挙します
     ① 入力は Simulinkへ部分時系列を入力: Configuration Parameter を採用



LSTM/GRU を実装する方法 (cont.)

② カスタム関数を用意し MATLAB Function ブロック内で間接的にネットワークを呼出

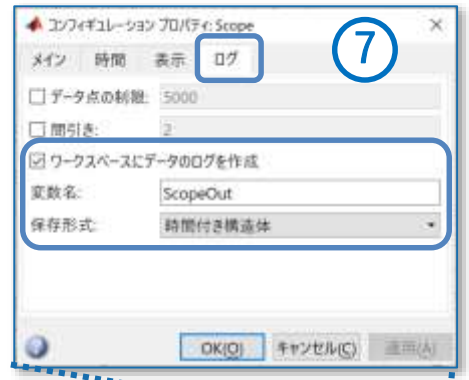
③ `coder.extrinsic` にてカスタム関数を指定しコード生成しないことを宣言

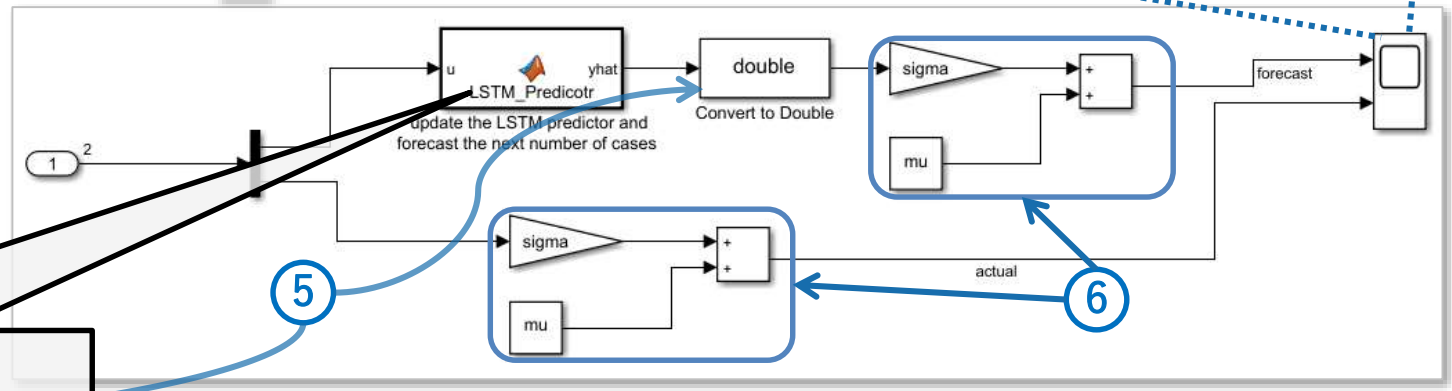
④ SeriesNetwork のインポートおよび更新は [事前学習済み学習器のインポートとエクスポート] を採用

⑤ カスタム関数からの返り値データ型を考慮

⑥ 元のスケールに変換

⑦ Scope ブロックをクリックし Configuration Property の「ログ」タグにある「ワークスペースにデータのログを作成」にチェックを入れ変数名を「ScopeOut」、保存形式を「時間付き構造体」へ編集





③

```
function yhat = LSTM_Predicotr(u)

yhat = single(0);
coder.extrinsic('my_update_predict_fcn');
yhat = my_update_predict_fcn(u);

end
```

②

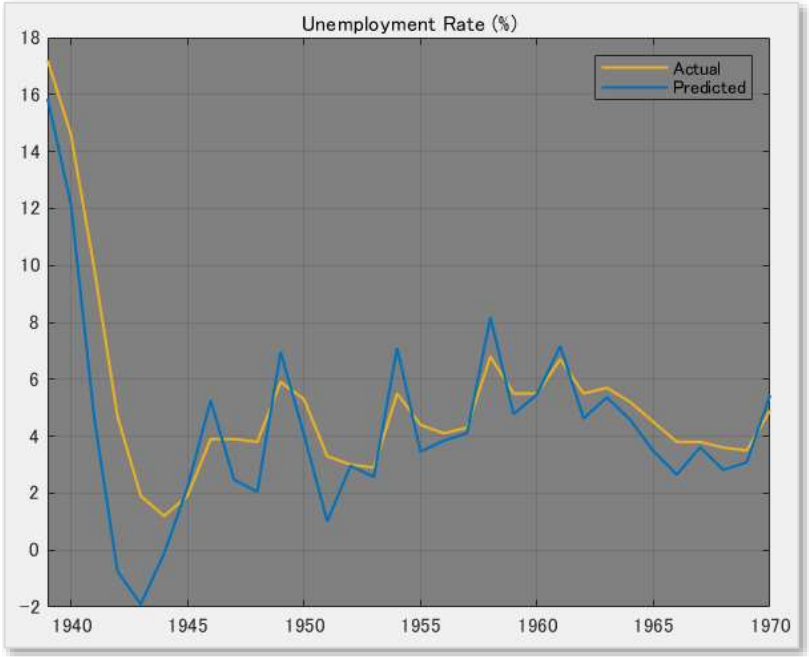
```
function yhat = my_udate_predict_fcn(y_t)

net = evalin('base', 'net');
[newnet, yhat] = predictAndUpdateState(net, y_t);
assignin('base', 'net', newnet);
end
```

④

状態空間モデルを実装する方法

1. サンプル「main_.mlx」について
   - 過去30年間の年次データおよび線型な状態空間モデルを使って
     翌年の失業率をパラメータ$(\alpha_1, \alpha_2, \beta)$を更新しながら推定

$$\begin{cases} \begin{bmatrix} x_t \\ x_{t-1} \end{bmatrix} = \begin{bmatrix} \alpha_1 & \alpha_2 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x_{t-1} \\ x_{t-2} \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \varepsilon_t^{(1)} \\ \varepsilon_t^{(2)} \end{bmatrix} \\ y_t - \beta \cdot z_t = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_t \\ x_{t-1} \end{bmatrix} + w_t \end{cases}$$



$x_t \coloneqq \Delta X_t = X_t - X_{t-1}$, X_t: (実際の)失業率

$z_t \coloneqq \Delta Z_t = \log(Z_t - Z_{t-1})$, Z_t: 名目GNP

参照) https://www.mathworks.com/help/econ/choose-state-space-model-specification-using-backtesting.html

状態空間モデルを実装する方法 (cont.)

2. 時刻付き部分時系列配列を作成

① 観測した時系列 $\{y_t\}$ に関する時刻付き部分時系列配列 **ObsUnemploymentRate** を作成

ObsUnemploymentRate =

t_1	y_1	y_2	y_3	$\cdots$	y_{w-1}	y_w
t_2	y_2	y_3	y_4	$\cdots$	y_w	y_{w+1}
t_3	y_3	y_4	y_5	$\cdots$	y_{w+1}	y_{w+2}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

② 時系列 $\{z_t\}$ に関する時刻付き部分時系列配列 **nGNP** を作成

nGNP =

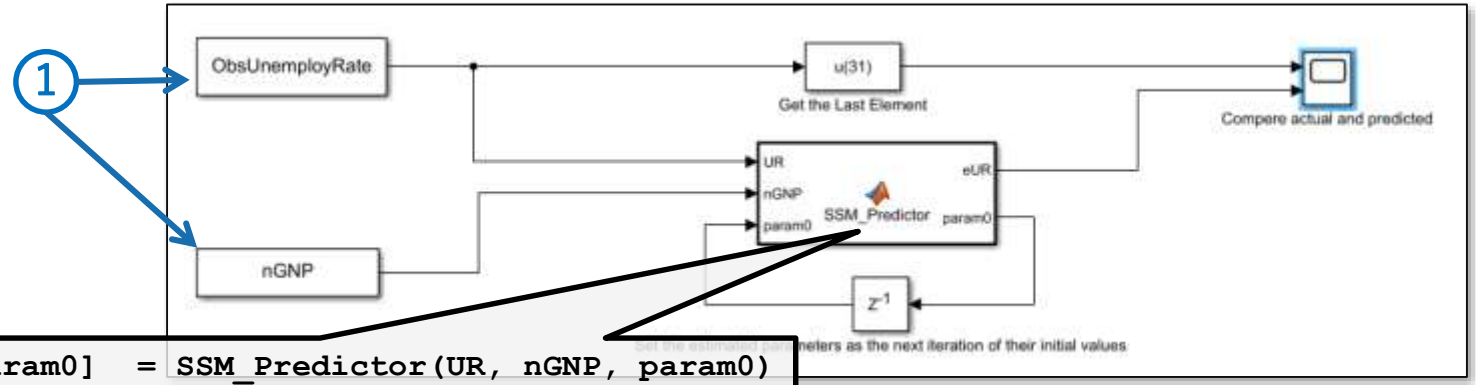
t_1	z_1	z_2	z_3	$\cdots$	z_{w-1}	z_w
t_2	z_2	z_3	z_4	$\cdots$	z_w	z_{w+1}
t_3	z_3	z_4	z_5	$\cdots$	z_{w+1}	z_{w+2}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

ただし、$z_u = \log(Z_{u+1} - Z_u)$, Z_u は名目GNP

状態空間モデルを実装する方法 (cont.)

2. モデル作成のポイント

- 「recursive_ssm_update.slx」を例にポイントを以下に列挙します
  - ① 入力は Simulinkへ部分時系列を入力: From Workspace ブロックを採用
  - ② カスタム関数を用意し MATLAB Function ブロック 内で間接的に未サポートのデータ型を呼出
  - ③ `coder.extrinsic` にてカスタム関数を指定しコード生成しないことを宣言
  - ④ カスタム関数からの返り値データ型が double であることを宣言



④
```
function [eUR, param0]  = SSM_Predictor(UR, nGNP, param0)
yhat = 0;
Z = diff(log(nGNP));
y = diff(UR);
coder.extrinsic('my_predict_fcn');
[yhat, param0] = my_predict_fcn(y, Z, param0);
eUR = yhat + UR(end);
end
```

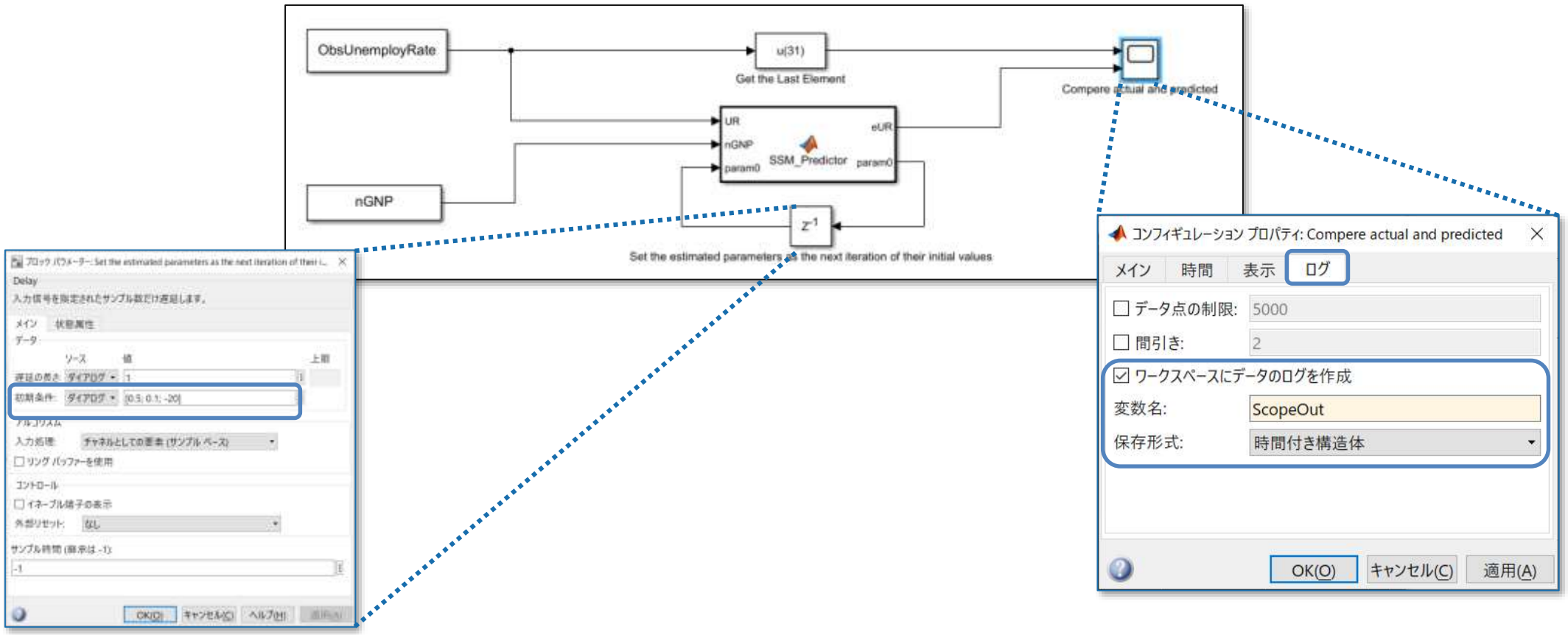
③ ②

```
function [yhat, param0] = my_predict_fcn(y,Z,param0)

Mdl = ssm(@(c)rwAR2ParamMap(c,y,Z));
[Mdl, param0] = estimate(Mdl, y, param0, 'Display', 'off');
yhat = forecast(Mdl, 1, y,'Predictors0',Z, ...
                    'PredictorsF',Z(end),'Beta',param0(end));
end
```

状態空間モデルを実装する方法 (cont.)

⑤ Delay ブロック パラメータの初期条件の値を [0.5; 0.1; -20] に設定

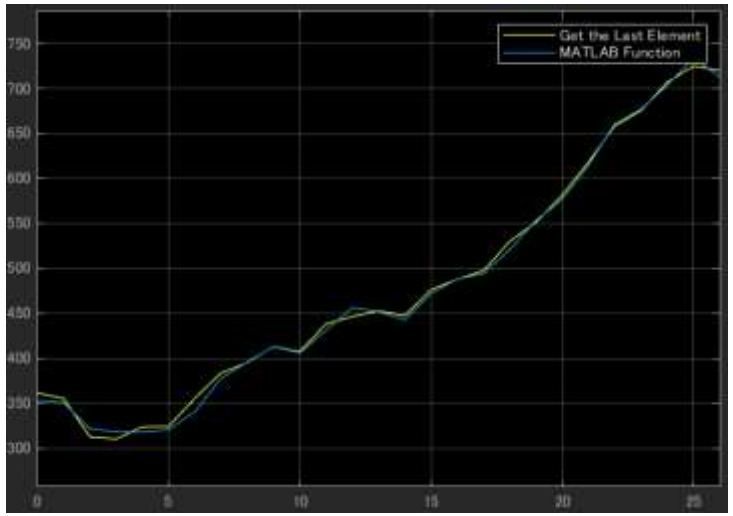
⑥ Scope ブロックをクリックし Configuration Property の「ログ」タグにある「ワークスペースにデータのログを作成」にチェックを入れ変数名を「ScopeOut」、保存形式を「時間付き構造体」へ編集



ARIMAX モデルを実装する方法

1. サンプル「ARIMAX¥time_series_regression.mlx」について
   - 過去30年間の年次データおよび ARIMAX(2,1,2) モデルを使って翌年の実質 GNP をパラメータ$(c, \varphi_1, \varphi_2, \psi_1, \psi_2, \beta_1, \beta_2, \beta_3)$を更新しながら推定

$$(1 - \varphi_1 L - \varphi_2 L^2)(y_t - y_{t-1}) = c + [\beta_1, \beta_2, \beta_3]\begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix} + (1 + \psi_1 L + \psi_2 L^2)\varepsilon_t$$



L は Lag Operator と呼ばれる作用素で $L^n y_t := y_{t-n}$

外生変数 X_1, X_2, X_3 は、それぞれ工業生産指数、雇用率および実質賃金の階差

参照） https://www.mathworks.com/help/econ/rolling-window-estimation-of-state-space-models.html
https://www.mathworks.com/help/econ/specify-presample-and-forecast-period-data-to-forecast-arimax-model.html

ARIMAX モデルを実装する方法 (cont.)

2. 時刻付き部分時系列配列を作成

① 対象である実質 GNP の時系列 $\{y_t\}$ から時刻付き部分時系列配列 **RealGNP** を作成

RealGNP =

t_1	y_1	y_2	y_3	$\cdots$	y_{w-1}	y_w
t_2	y_2	y_3	y_4	$\cdots$	y_w	y_{w+1}
t_3	y_3	y_4	y_5	$\cdots$	y_{w+1}	y_{w+2}
:	:	:	:	:	:	:

② ARMAIX(2,1,2) つまり ARの次数が「2」階差「1」であることから $\{y_{t-3}, y_{t-2}, y_{t-1}\}$ に相当する時刻付き部分時系列配列**PreRealGNP** を作成

PreRealGNP =

t_1	y_1	y_2	y_3
t_2	y_2	y_3	y_4
t_3	y_3	y_4	y_5
:	:	:	:

ARIMAX モデルを実装する方法 (cont.)

③ 外生変数である工業生産指数、雇用率および実質賃金のそれぞれの階差から時刻付き部分時系列配列を作成

$X_n =$

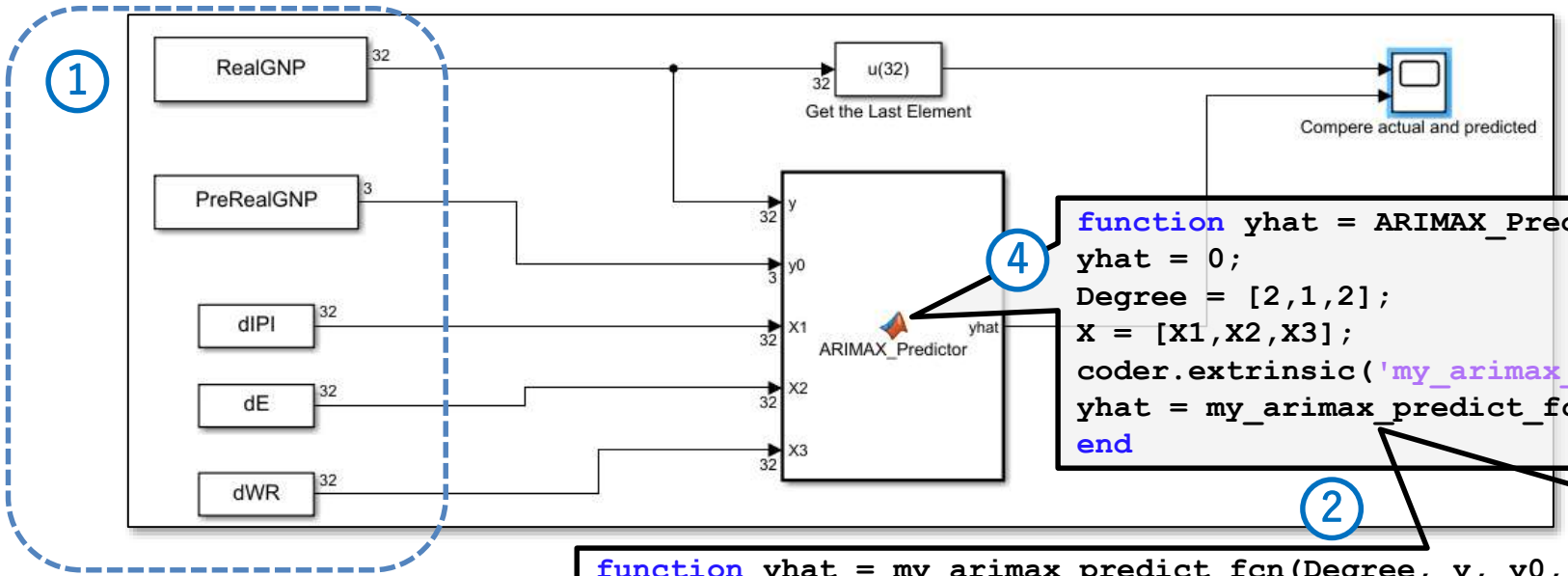
t_1	$x_{n,1}$	$x_{n,2}$	$x_{n,3}$	$\cdots$	$x_{n,w-1}$	$x_{n,w}$
t_2	$x_{n,2}$	$x_{n,3}$	$x_{n,4}$	$\cdots$	$x_{n,w}$	$x_{n,w+1}$
t_3	$x_{n,3}$	$x_{n,4}$	$x_{n,5}$	$\cdots$	$x_{n,w+1}$	$x_{n,w+2}$
:	:	:	:	:	:	:

$x_{1,u}$：工業生産指数の階差
$x_{2,u}$：雇用率の階差
$x_{3,u}$：実質賃金の階差

$\mathbf{dIPI} = X_1$
$\mathbf{dE} = X_2$
$\mathbf{dWR} = X_3$

ARIMAX モデルを実装する方法 (cont.)

3. モデル作成のポイント
   - 「recursive_ssm_update.slx」を例にポイントを以下に列挙します
     ① 外生変数を含めデータの入力は Simulinkへ部分時系列を入力: From Workspace ブロックを採用
     ② カスタム関数を用意し MATLAB Function ブロック 内で間接的に未サポートのデータ型を呼出
     ③ coder.extrinsic にてカスタム関数を指定しコード生成しないことを宣言
     ④ カスタム関数からの返り値データ型が double であることを宣言




```
function yhat = ARIMAX_Predictor(y, y0, X1, X2, X3)
yhat = 0;
Degree = [2,1,2];
X = [X1,X2,X3];
coder.extrinsic('my_arimax_predict_fcn');
yhat = my_arimax_predict_fcn(Degree, y(1:end-1), y0, X);
end
```

```
function yhat = my_arimax_predict_fcn(Degree, y, y0, X)
Mdl = arima(Degree(1), Degree(2), Degree(3));
Mdl = estimate(Mdl,y,'Y0',y0,'X',X(1:end-1,:),'Display','off');
yhat = forecast(Mdl,1,y(end-p-D-1:end),'X0',X(end-2:end-1,:),'XF',X(end,:));
end
```

ARIMAX モデルを実装する方法 (cont.)

⑤ Scope ブロックをクリックし Configuration Property  の「ログ」タグにある「ワークスペースにデータのログを作成」にチェックを入れ変数名を「ScopeOut」、保存形式を「時間付き構造体」へ編集