DASH AND PLOTLY FOR INTERACTIVE PLOTTING

Tutorial with a case-study of the Bifurcation Diagram

Kevin Pouget, Red Hat SPICE Senior Software Engineer

February 19, 2020

DASH+PLOTLY

Presentation

Dash

`https://plot.ly/dash/`

- "Dash is the fastest way to build interactive analytic apps" (says their website)
- Open source under MIT licensing
- Dash is available for both Python and R (similar to RStudio)
  - good&illustrated documentation: `https://dash.plot.ly/`

Plotly

- HTML/JS/SVG plotting from Python
- many ways to customize graphs
- works with or without Dash
  - good&illustrated documentation: `https://plot.ly/python/`

```
pip install --user dash==1.8.0 # installs plotly as well
```

DASH LAYOUT

HTML … in Python

```
import dash_html_components as html
app.layout = html.Div(children=[
    html.H1(
        children='Hello Dash',
        style={'textAlign': 'center',
               'color': colors['text']}),

    html.Div(
        id='my-div',
        children='Dash: A web app framework for Python.',
        style={'textAlign': 'center',
               'color': colors['text']
    }),
])
```

DASH LAYOUT

HTML … in Python … plus complex components

```
import dash_core_components as dcc
dcc.Dropdown(value='MTL', options=[
                {'label': 'New York City', 'value': 'NYC'},
                {'label': 'Montréal', 'value': 'MTL'},
                {'label': 'San Francisco', 'value': 'SF'}])

dcc.Checklist(...), dcc.RadioItems(...)
dcc.Slider(min=0, max=9, value=5)

dcc.Tabs(value='tab-1-example', children=[
        dcc.Tab(label='tab one', value='tab-1-example'),
        dcc.Tab(label='tab two', value='tab-2-example')])

dcc.Graph(id='example-graph-2', figure={'data': [...], 'layout':
```

DASH CALLBACKS

HTML … in Python … plus complex components and callbacks!

```
@app.callback(
    Output('my-div', 'children'),
    [Input('my-slide-id', 'value')])
def update_output(slide_value):
    return f"You've entered '{slide_value}'"
```

- when the `value` of `my-slide-id` changes,
- then `update_outout(value)` gets called.
- and its return value replaces `my-div`'s `children`.

DASH CALLBACKS

HTML … in Python … plus complex components and callbacks!

```
@app.callback(
    Output('my-div', 'children'),
    [Input('my-slide-id', 'value')])
def update_output(slide_value):
    return f"You've entered '{slide_value}'"
```

```
@app.callback(
    Output('my-graph', 'figure'),
    [Input('my-slide-id', 'value')])
def update_graph(slide_value):
    return go.Figure(data=go.Scatter(x=[...]), y=[...]))
```

DASH CALLBACKS

you can get Dash callbacks from ...

- button clicks, text (`Div/P`) clicks
- dropdown list value entered/changed
- graph hover/click on value
- period timers, URL address change, ...

DASH CALLBACKS

you can get Dash callbacks from ...

- button clicks, text (`Div`/`P`) clicks
- dropdown list value entered/changed
- graph hover/click on value
- period timers, URL address change, ...

from Dash callbacks, you can ...

- update `input` values
- generate new HTML elements
- update the CSS style of layout HTML elements
- generate any kind of plotly graph

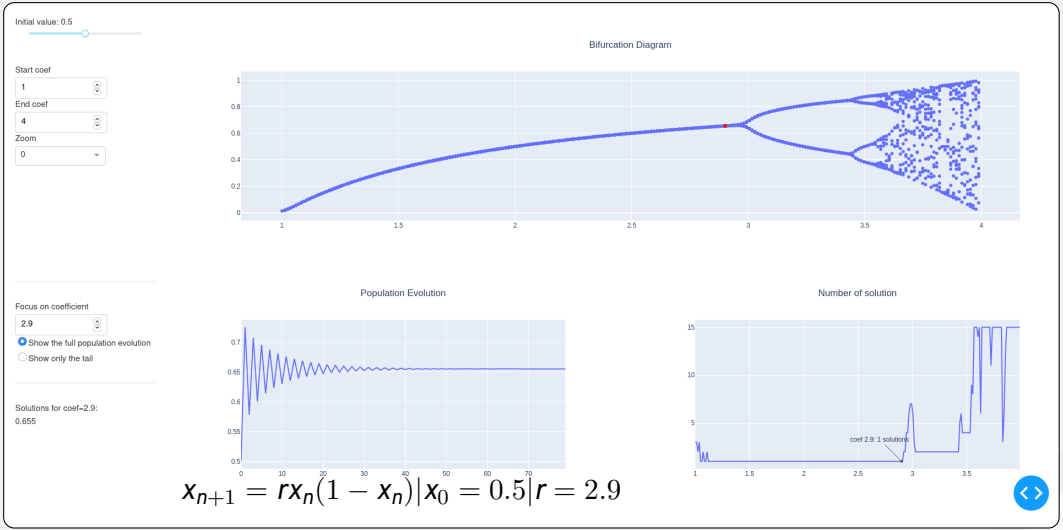
DASH CALLBACKS

from Dash callbacks, you CANNOT ...

- use global variables to store anything
  - multi-process WSGI backends might bite you :)
- set callbacks to generated HTML elements
  - callbacks+inputs/outputs must be defined at app creation time
  - workaround: use CSS style to hide/show elements instead (`display: none`)
- have more than one callback updating a given property
  - You have already assigned a callback to the output with ID "..." and property "...". An output can only have a single callback function. Try combining your inputs and callback functions together into one function.
- have dependency cycles (X generates Y, Y generates X)

CASE STUDY: BIFURCATION DIAGRAM

CASE STUDY: BIFURCATION DIAGRAM

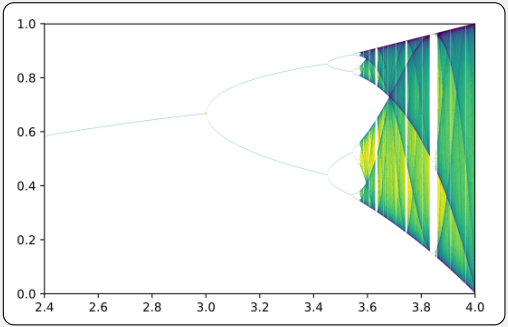


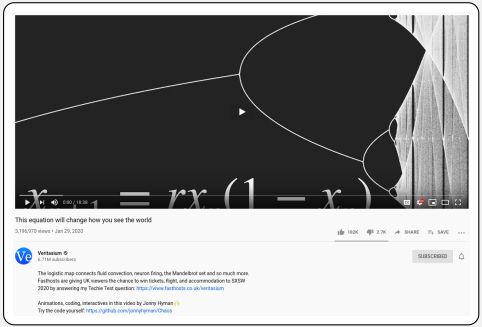
$$x_{n+1} = rx_n(1 - x_n) | x_0 = 0.5 | r = 2.9$$

git clone `https://github.com/kpouget/bifurq`

Red Hat

CASE STUDY: BIFURCATION DIAGRAM



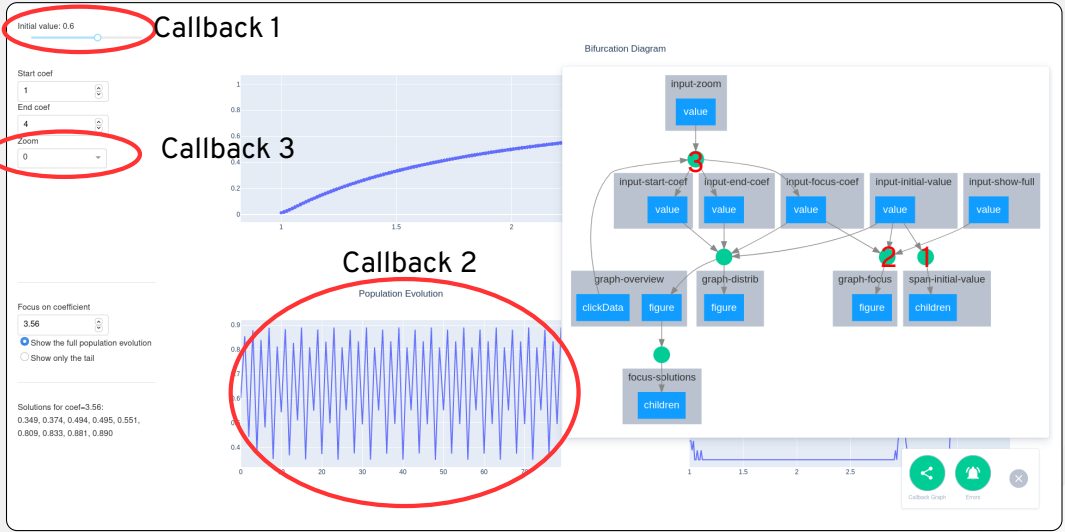


- Case-study repository (check the branches)
  - `https://github.com/kpouget/bifurq`
- Veritasium video *"This equation will change how you see the world"*
  - `https://www.youtube.com/watch?v=ovJcsL7vyrk`
- Matplotlib Bifurcation diagram my Morn, Creative CC BY SA

`https://en.wikipedia.org/wiki/File:Logistic_Map_Bifurcation_Diagram,_Matplotlib.svg`

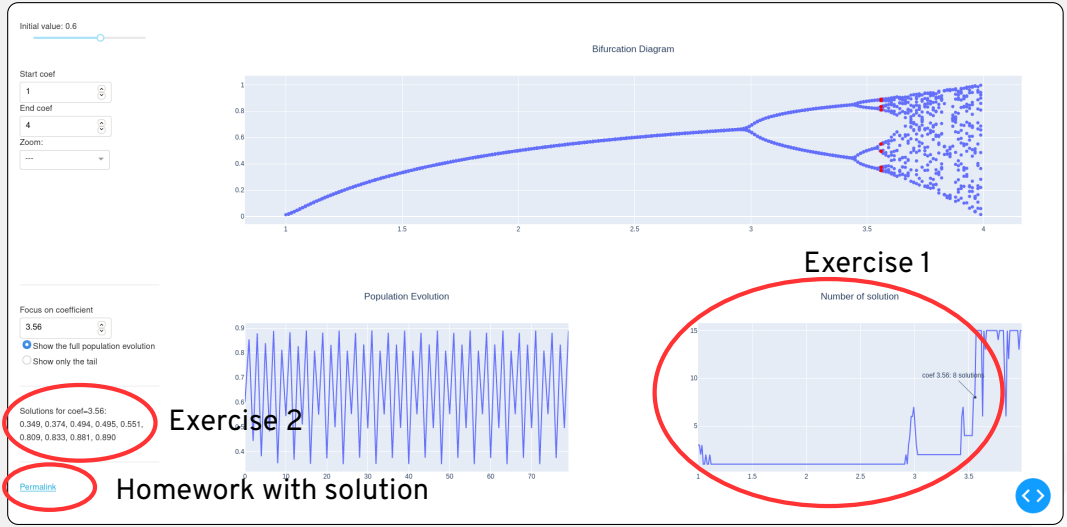
git clone `https://github.com/kpouget/bifurq`

CASE STUDY: BIFURCATION DIAGRAM



git clone `https://github.com/kpouget/bifurq`

CASE STUDY: BIFURCATION DIAGRAM

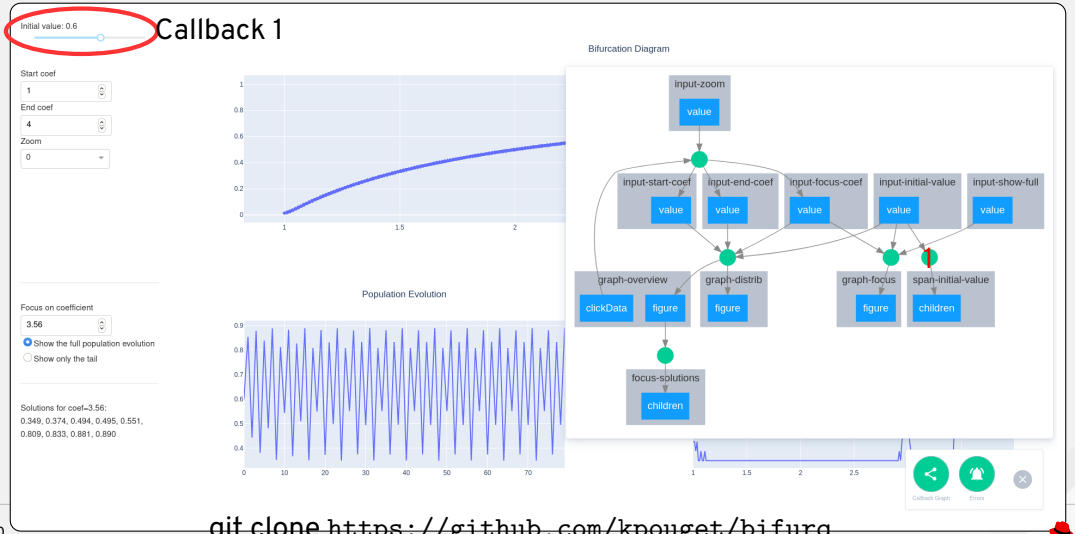


git clone `https://github.com/kpouget/bifurq`

Red Hat

DASH CALLBACK 1

Update initial-value label




```
git clone https://github.com/kpouget/bifurq
```


Red Hat

DASH CALLBACK 1

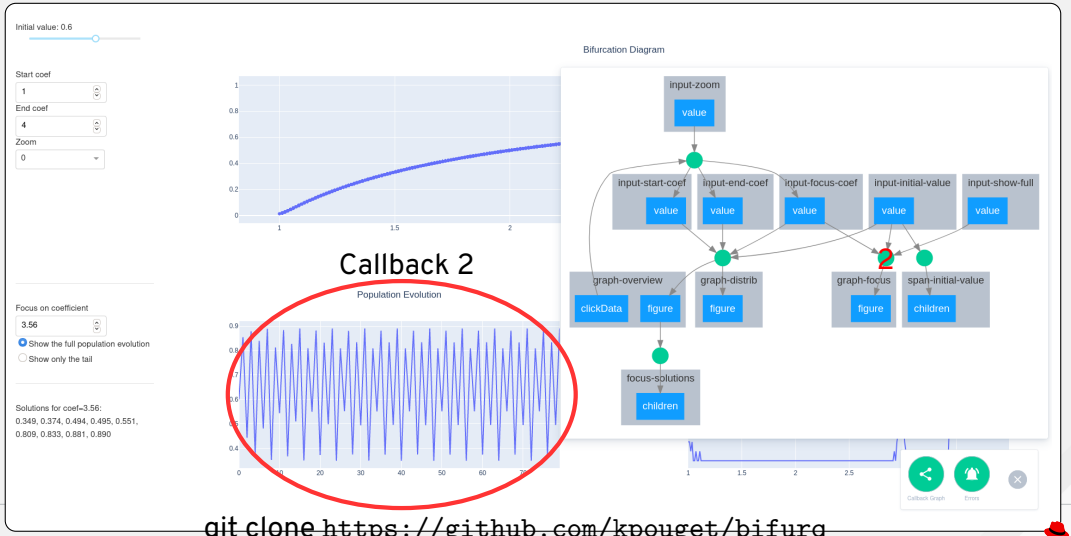
Update initial-value label

```
@app.callback(
    Output('span-initial-value', 'children'),
    [Input('input-initial-value', 'value')])
def update_initial_value(value):
    return str(value) # or f"{value*100:.0f}%"
```


git clone `https://github.com/kpouget/bifurq`

DASH CALLBACK 2

Draw focus graph



git clone `https://github.com/kpouget/bifurq`

Red Hat

DASH CALLBACK 2

Draw focus graph

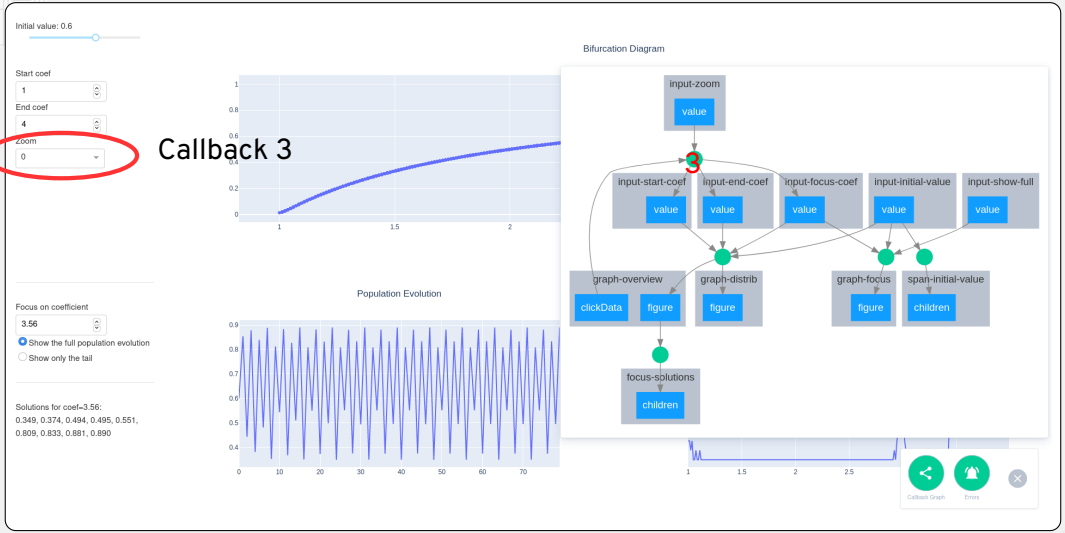
```
@app.callback(
    Output('graph-focus', 'figure'),
    [Input('input-initial-value', 'value'),
     Input('input-focus-coef', 'value'),
     Input('input-show-full', 'value')]) # on/off actually
def draw_focus(init_value, coef, full):
    x = range(N_COMPUTE) if full else \
        range(N_COMPUTE - KEEP, N_COMPUTE)
    y = compute_evolution(init_value, coef, full=full)

    fig = go.Figure(data=go.Scatter(x=list(x), y=y))
    fig.update_layout(title={'text': "Population Evolution"})

    return fig
```

git clone `https://github.com/kpouget/bifurq`

DASH CALLBACK 3: ZOOM ON COEF



git clone `https://github.com/kpouget/bifurq`

Red Hat

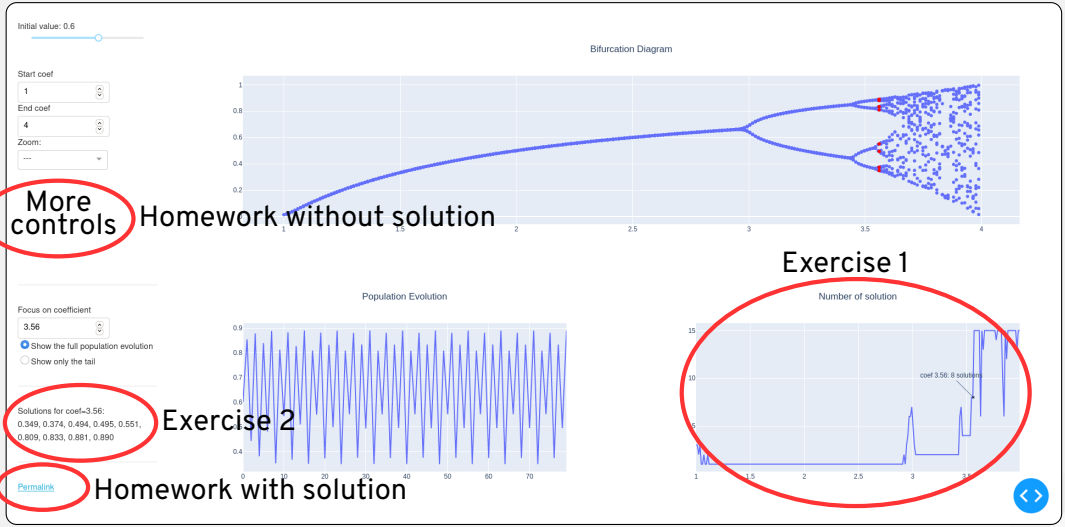
DASH CALLBACK 3: ZOOM ON COEF

```
@app.callback(
    [Output('input-start-coef', 'value'),
     Output('input-end-coef', 'value'),
     Output('input-focus-coef', 'value')],
    [Input('input-zoom', 'value'),
     Input('graph-overview', 'clickData')])
def update_coef(zoom, clickData):
    trigger = dash.callback_context.triggered[0]["prop_id"]
    if trigger.startswith('graph-overview'):
        # triggered by click on graph-overview point
        return [no_update]*2, clickData['points'][0]['x']

    # triggered by zoom-input value changed
    try: return ZOOMS[zoom]
    except KeyError: return START_COEF, END_COEF, FOCUS_COEF
```


git clone `https://github.com/kpouget/bifurq`

CASE STUDY: EXERCISES



git clone `https://github.com/kpouget/bifurq`

CASE STUDY: EXERCISES

Exercise 1: ‘Number of solution’ diagram

- add `dcc.Graph` in the layout
- add `Output(id, 'figure')` in the `draw_overview` callback
- build `go.Figure(data=[go.Scatter(x=count_x, y=count_y)])`
- add `go.layout.Annotation` text annotation

Exercice 2: ‘solutions for coef’ text

- add `html.Div` in the layout
- new callback with `Input('graph-overview', 'figure')`
- add state info `State('input-focus-coef', 'value')`
- build text with `solutions = graph['data'][1]['y']`
  - (the solutions are already computed and plotted in red in the 2nd graph figure)

git clone `https://github.com/kpouget/bifurq`

CASE STUDY: PERMALINK HOMEWORK

Key feature, but not so easy to build ...

```
@app.callback(Output('permalink', 'href'),
[Input(f"input-{input}", 'value') for input in INPUT_NAMES])
def get_permalink(*args):
    return "?"+"&".join(f"{k}={v}" \
           for k, v in zip(INPUT_NAMES, map(str, args)))
```

```
@app.callback(Output('page-content', 'children'),
              [Input('url', 'search')])
def display_page(search):
    search_dict = urllib.parse.parse_qs(search[1:])
    return build_layout(search_dict)
```

git clone `https://github.com/kpouget/bifurq`