

MLDL

Machine Learning and Deep Learning Conference 2021

A Numerical Compact Photocurrent Model and its Implementation via Xyce-PyMi

- Presenter: Joshua Hanson (1356)
- Team members: Biliana Paskaleva (1356), Pavel Bochev (1400), Paul Kuberry (1442), Chuck Hembree (1344), Eric Keiter (1355)
- Funding: REHEDS LDRD

Overview and Motivation



- **Motivation:** Photocurrents generated within semiconductor devices that are exposed to ionizing radiation may adversely affect circuit performance.
- **Objective:** We want to predict **photocurrent** as a function of radiation **dose rate**, **voltage(s)** across the device, and **time**, dynamically within a circuit simulation.
- **State of the art** compact photocurrent models apply **physical assumptions and approximations** with limited validity to render physics models solvable in closed-form.
- **Conjecture:** **numerical order reduction techniques** can provide a viable alternative to analytic approximations.
- **In this work** we consider a *pn*—diode exposed to a spatially uniform radiation pulse.

Development of traditional compact photocurrent models

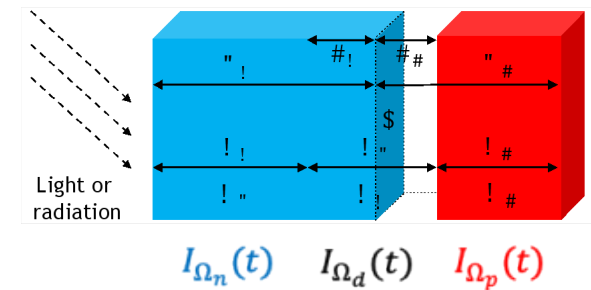
Compact analytic photocurrent models are developed by treating separately different device regions.

- Motivated by the need to **simplify the physics model** to render it **solvable in closed form**.

Step 1: Split device into a depletion region and quasi-neutral P and N-regions:

Carriers in Ω_d are **quickly converted to prompt photocurrent** (strong electric field).

Carriers in Ω_p, Ω_n gradually drift/diffuse into Ω_d : produce **delayed photocurrents**.



Step 2: Represent total photocurrent as sum of prompt and delayed photocurrents:

$$I(t) = I_{\Omega_d}(t) + I_{\Omega_p}(t) + I_{\Omega_n}(t) \quad \text{Model prompt photocurrent as: } I_{\Omega_d}(t) = qAw_d(V(t))g(t)$$

Model $I_{\Omega_p}(t)$ and $I_{\Omega_n}(t)$ assuming **charge neutrality, congruence, low dose rate** approximations:

$$w_d(V) = \sqrt{\frac{2\epsilon}{q} \left(\frac{1}{N_A} + \frac{1}{N_D} \right) \max(0, V_{bi} - V)}$$

Drift-Diffusion \rightarrow Ambipolar Diffusion Equation (ADE):
$$\frac{\partial \delta p}{\partial t} = D_p \frac{\partial^2 \delta p}{\partial x^2} - \mu_p E_n \frac{\partial \delta p}{\partial x} - \frac{\delta p}{\tau_p} + g(x, t), \quad (x, t) \in \Omega_n \times T$$

Using homogeneous BC:
$$I_n(t) = qAD_p \frac{\partial \delta p}{\partial x}(w_n, t) + \mu E_n \delta p(w_n, t)$$

Projection-based model order reduction

Discretize the 1D ADE using finite elements: $\frac{\partial \delta p}{\partial t} = D_p \frac{\partial^2 \delta p}{\partial x^2} - \mu_p E_n \frac{\partial \delta p}{\partial x} - \frac{\delta p}{\tau_p} + g(x, t), \quad (x, t) \in \Omega_n \times T$



Full-Order Model (FOM):

$$\begin{aligned} \dot{\mathbf{u}}(t) &= (A + E(t)N)\mathbf{u}(t) + \mathbf{g}(t) \\ \mathbf{u}(t) &:= (u_1(t), \dots, u_{m-1}(t)) \in \mathbb{R}^{m-1} \end{aligned} \quad u_{\text{ADE}}(x, t) = \sum_{i=1}^{m-1} u_i(t)v_i(x)$$



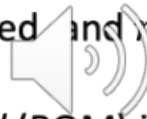
- $\frac{\partial \delta p}{\partial x}(w_n, t)$ in the boundary current computed via gradient of shape functions: $\frac{\partial \delta p}{\partial x}(w_n, t) \leftarrow \sum_{i=1}^{m-1} u_i(t) \frac{dv_i}{dx}(w_n)$
- Can we find a change of coordinates so that only a small number of u_i 's in the FOM are "active" and most are approximately zero (so that they can be discarded)?
- We wish to project the high-dimensional state $\mathbf{u} \in \mathbb{R}^{m-1}$ onto a low-dimensional hyperplane: $\tilde{\mathbf{u}} := P\mathbf{u} \in \mathbb{R}^r$
- With P determined, we can compute the pushforward dynamics for $\tilde{\mathbf{u}}$ (i.e., an r -dimensional system of ODEs).

Projection-based model order reduction

- How do we identify P ? Truncated SVD:

$$X = \begin{bmatrix} | & & | \\ \mathbf{u}_1 & \cdots & \mathbf{u}_q \\ | & & | \end{bmatrix} = [\tilde{U} \quad \tilde{U}_{\text{trun}}] \begin{bmatrix} \tilde{\Sigma}^T & 0 \\ 0 & \tilde{\Sigma}_{\text{trun}}^T \end{bmatrix} \begin{bmatrix} \tilde{V}^T \\ \tilde{V}_{\text{trun}}^T \end{bmatrix} \approx \tilde{U} \tilde{\Sigma} \tilde{V}^T$$

where $0 < r \leq m - 1$ modes are preserved and $m - 1 - r$ modes are truncated.

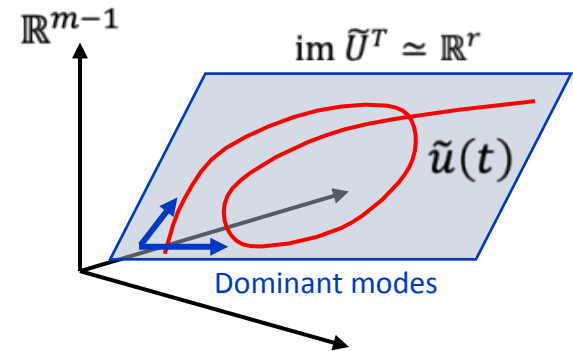


- Now $P = \tilde{U}^T$ and the *reduced-order model* (ROM) is:

$$\begin{aligned} \dot{\tilde{\mathbf{u}}}(t) &= (\tilde{A} + \mathbf{E}(t)\tilde{B})\tilde{\mathbf{u}}(t) + \tilde{U}^T \mathbf{g}(t) \\ \tilde{\mathbf{u}}(0) &= \tilde{U}^T \mathbf{u}(0) \\ \mathbf{u}(t) &\approx \tilde{U} \tilde{\mathbf{u}}(t) \end{aligned}$$

where $\tilde{A} = \tilde{U}^T A \tilde{U} \in \mathbb{R}^{r \times r}$ and $\tilde{B} = \tilde{U}^T B \tilde{U} \in \mathbb{R}^{r \times r}$

For an effective ROM, we want: $r \ll m - 1$



- ADE FOM defined by partitioning Ω_n and Ω_p each into 1024 uniform elements: $\dim(\text{FOM}) = 2048$.
- We obtain excellent accuracy with just a few dominant modes: $r \sim 5$.

Model Implementation

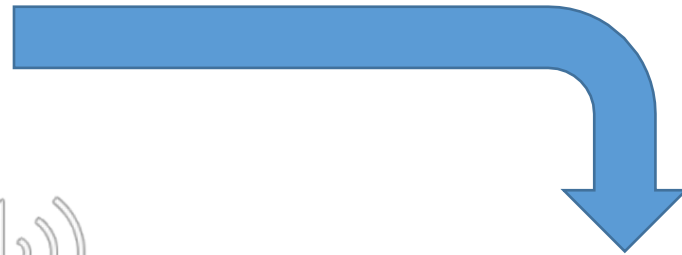
$$\dot{\mathbf{u}}(t) = \mathbf{A}\mathbf{u}(t) + \mathbf{E}(t)\mathbf{N}\mathbf{u}(t) + \mathbf{B}g(t)$$

$$I(t) = \mathbf{C}\mathbf{u}(t) + w_D(V(t))Dg(t)$$

$$E(t) \propto V_{bi} - V(t)$$

$$w_D(V(t)) \propto \sqrt{\max(0, V_{bi} - V(t))}$$

Translate math to Python code



- The ROM parameters are the **reduced matrices**, which could be stored in a pickle file, .csv, .txt, etc.
- The FOM parameters are **calibrated to experiment via TCAD** (or "quasi-TCAD"), then the ROM matrices are extracted automatically.

```
# Open loop dynamics
def forward(self, t, x, g, V):
    return np.matmul(self.A, x) + 0.5 * (V - self.V_bi) * np.matmul(self.N, x) + self.B * g

# Open loop output
def output(self, t, x, g, V):
    return np.dot(self.C, x) + np.sqrt(np.maximum(0, V - self.V_bi)) * self.D * g

# Closed loop dynamics/output for a given radiation input (descriptor protocol)
def generate_closed_loop(self, input):

    def forward_CL(self, t, x, V):
        return self.forward(t, x, input(t), V)
    self.forward_CL = forward_CL.__get__(self)

    def output_CL(self, t, x, V):
        return self.output(t, x, input(t), V)
    self.output_CL = output_CL.__get__(self)
```

Model Implementation

1) Model form definition
(translate math to code):

```

4 class ADE_ROM:
5     def __init__(self, **kwargs):
        ...
46 # Open loop dynamics
47 def forward(self, t, x, g, V):
48     return np.matmul(self.A, x) +
49
50 # Open loop output
51 def output(self, t, x, g, V):
52     return np.dot(self.C, x) + np.

```



2) Instantiate model
(read in pre-calibrated reduced matrices):

```

1 import numpy as np
2 import pickle
3 import dynamic_models
4
5 # Import matrices
6 with open('Models/ROM_n4p1_matrices.pkl', 'rb') as file:
7     An, Ap, Nn, Np, Bn, Bp, Cn, Cp, D = pickle.load(file)
8
9 # Initialize model
10 ROM = dynamic_models.ADE_ROM(n_dim = 4, p_dim = 1)
11 ROM.An = An
12 ROM.Nn = Nn
13 ROM.Bn = Bn

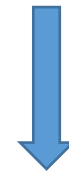
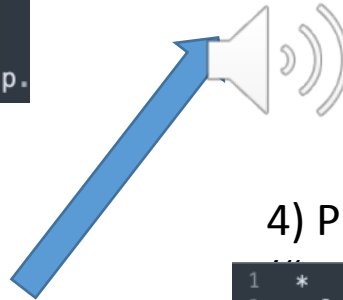
```

3) Import model instance into Xyce-PyMi
custom device definition template:

```

15 class Device(BaseDevice):
16
17     def processPythonParams(self, b_params,
        ...
88     def computeXyceVectors(self, fSV, solV, stoV, staV, deviceOpt
89         origFlag, F, Q, B, dFdX, dQdX, dFdXdVp, dQdXdVp,
90         b_params, d_params, i_params, s_params):

```



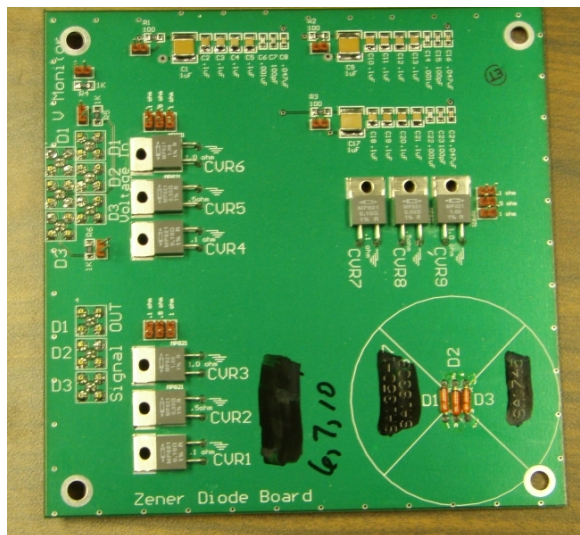
4) Plug radiation model into netlist

```

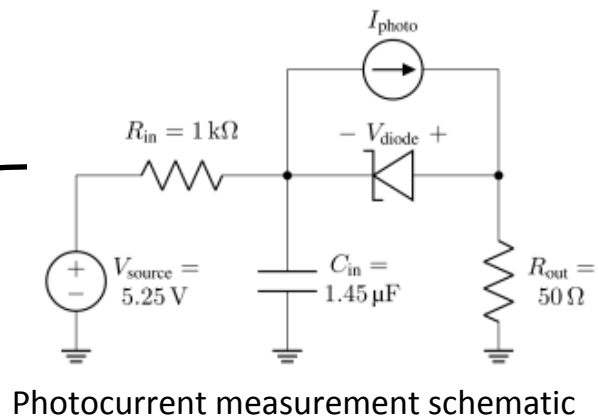
1 *
2 * SIMULATION DIRECTIVES
3 *
4 .TRAN 1.0E-9 14u 0u 0.5E-8
5 .PRINT TRAN I(Rout)
6 *
7 * CIRCUIT NETLIST
8 *
9 Vin      2      0      5.25
10 Rin     2      1      1k
11 Cin     1      0      1.45u
12 Rout    3      0      50
13 Xdiode   3      1      MMSZ5236BT1
14 YGENEXT Iphoto 1      3
15 + SPARAMS={NAME=MODULENAME,MODELFILE,DATAFILE VALUE=RadModels.py,Inst_ADE_ROM.

```

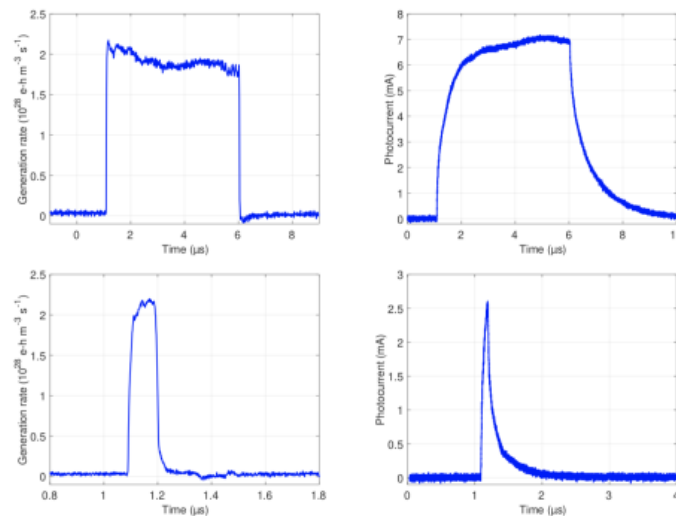
Photocurrent experimental set up



Same circuit recreated in Xyce netlist for simulations



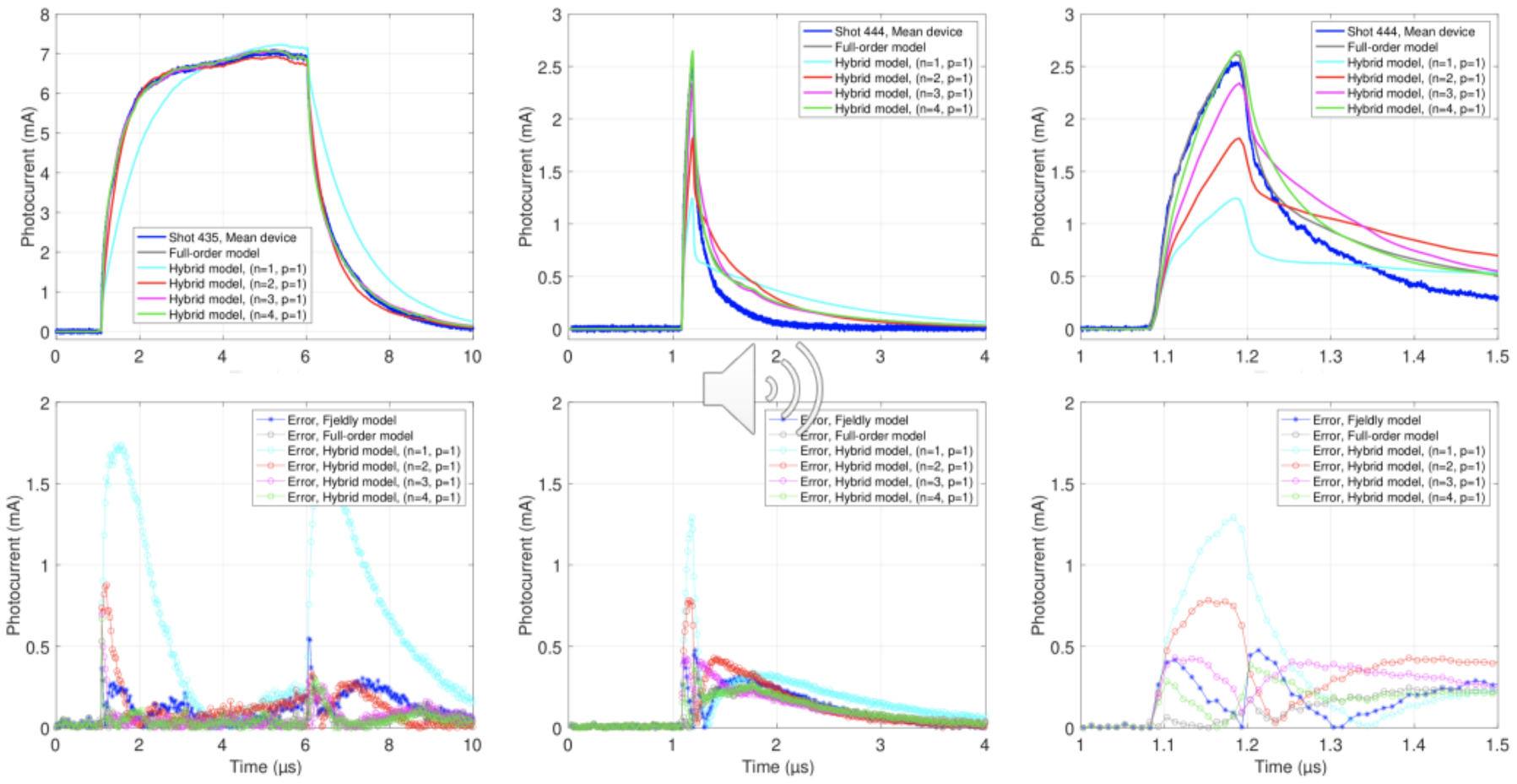
Shot #	How used	TD (rad)	PW (s)
435	calibration	2250.9	4.95e-6
436	prediction	2272.3	4.95e-6
444	calibration	53.0	1.05e-7
445	prediction	53.5	1.04e-7



Total dose (TD) and pulse width (PW) for the LMTF LINAC Shots

Sample radiation pulses (left), and measured photocurrent (right)

Photocurrent prediction results



FOM: system of $1023 \times 2 = 2046$ ODEs

ROM_{n=3,p=1} - system of $3+1=4$ ODEs $\Rightarrow 2046/4=511:1$ compression ratio

ROM_{n=4,p=1} - system of $4+1=5$ ODEs $\Rightarrow 2046/5 \sim 409:1$ compression ratio

Conclusions and Follow-Up Work



- **Numerical compression strategies** for compact photocurrent modelling can circumvent the need for simplifying analytic assumptions and approximations, yielding a “faithful-to-the-physics” model which is fast to develop and easy to implement.
- **Reduced development time:** ROM extraction from full-order models (FOM) is “automated.”
 - Main effort can be concentrated on the [development of FOMs](#).
- **Extraction of compact models** from FOMs has multiple advantages:
 - Compact models and FOMs [reference the same physics](#).
 - Reduces remodeling and produces [more generalizable, more credible](#) compact models.
 - These models require [less-per-instance calibration](#):
 - Reduced [calibration data gathering, formatting, and storage burdens](#).
 - Accelerated end-to-end ModSym process.
- **Follow-up work:**
 - Apply projection-based ROM in combination with polynomialization to [more accurate physics models](#) (e.g., drift-diffusion).
 - Use [simple physics-based descriptions](#) (which can achieve high compression ratios) and augment the model with perturbations (e.g., neural nets) to achieve a close fit to experimental measurements.