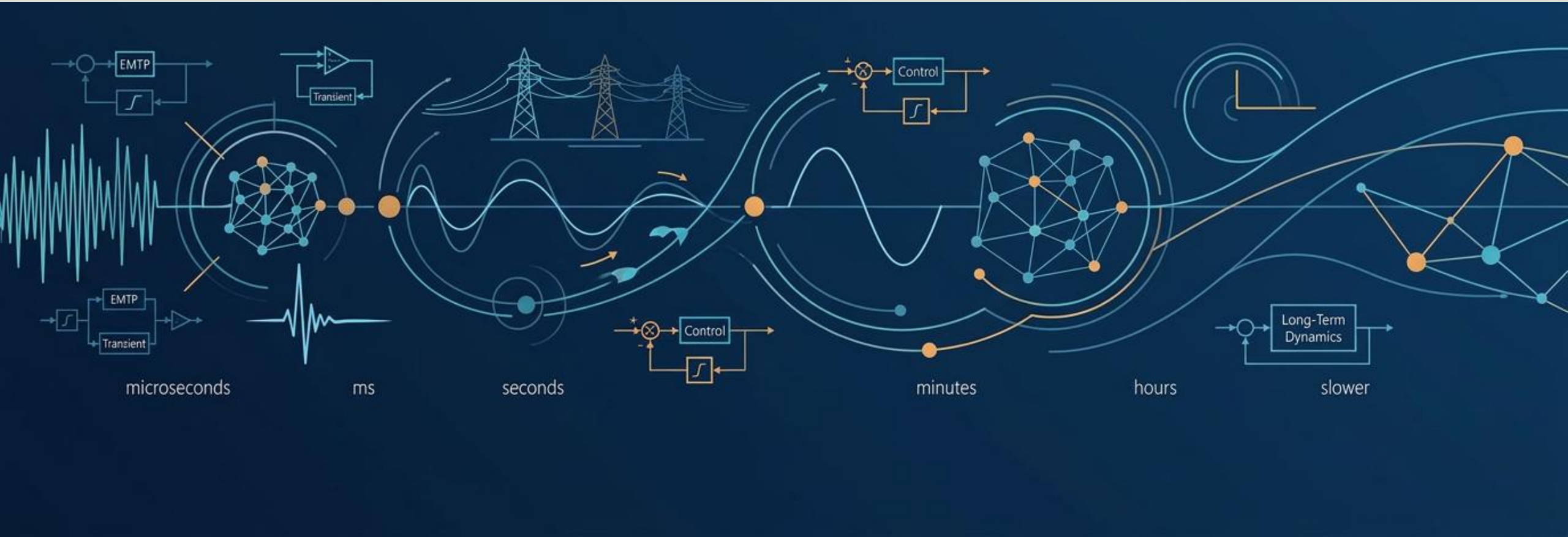
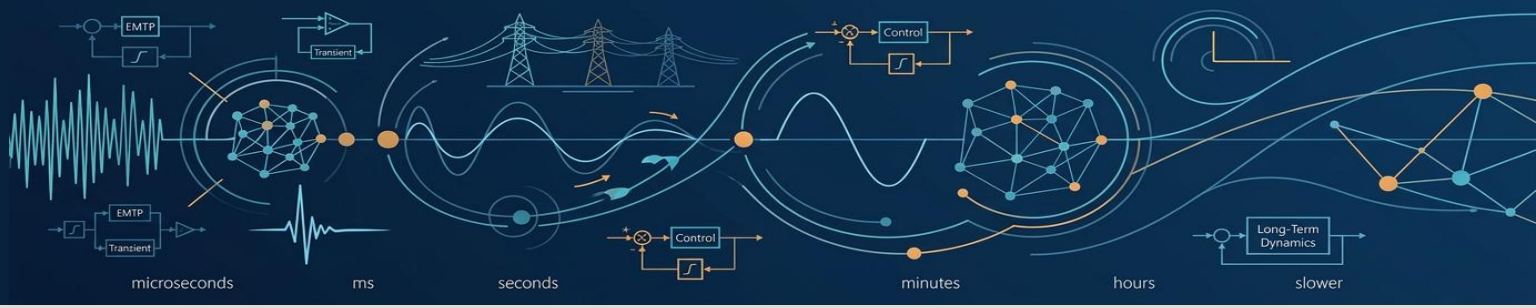
Projektovanje pomoću računara u elektroenergetici

Prof. dr Goran Dobrić
Prof. dr Mileta Žarković

Vremenske konstante i tipovi simulacija u elektroenergetskim sistemima

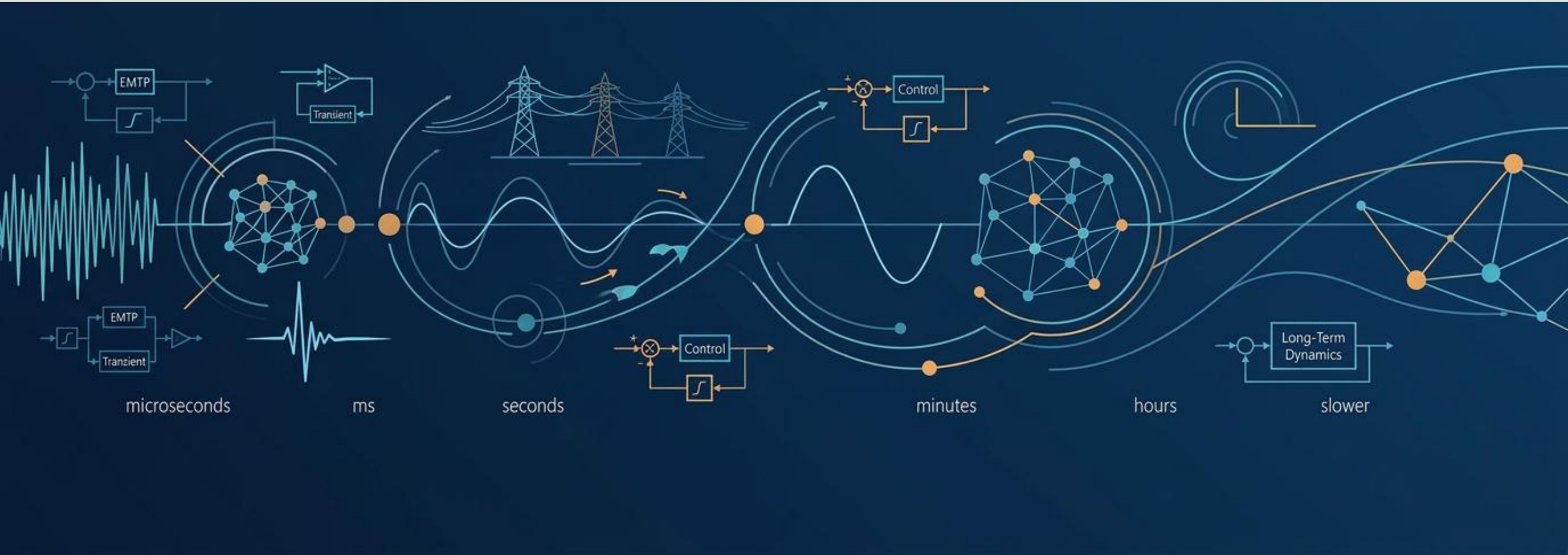


Ciljevi predavanja

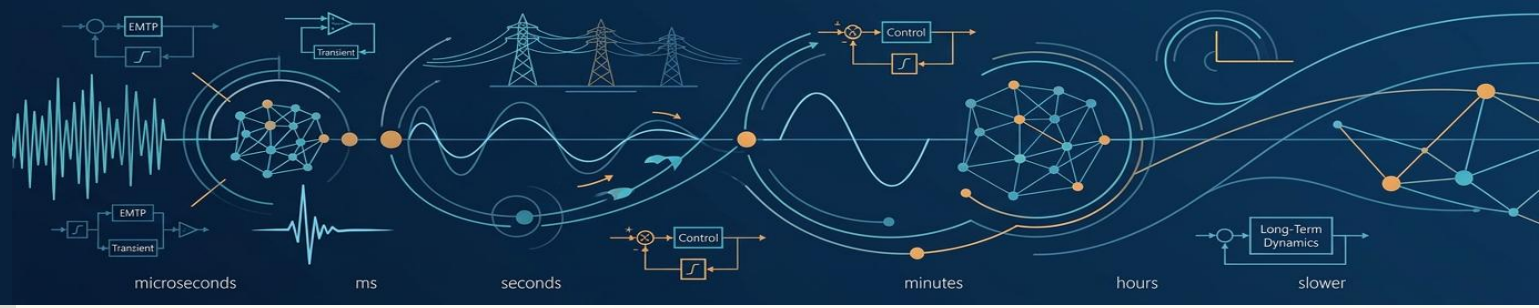


- Razumeti spektar vremenskih skala u elektroenergetici (mikrosekunde do minuti)
- Povezati vremenske konstante sa tipovima matematičkih jednačina (algebarske, ODE, DAE)
- Naučiti kako korak simulacije utiče na tačnost i računsko vreme
- Upoznati se sa konceptom real-time simulacija i OPAL-RT platformom
- Videti praktične Matlab primere koji ilustruju ove koncepte

1. Vremenske skale u elektroenergetskim sistemima



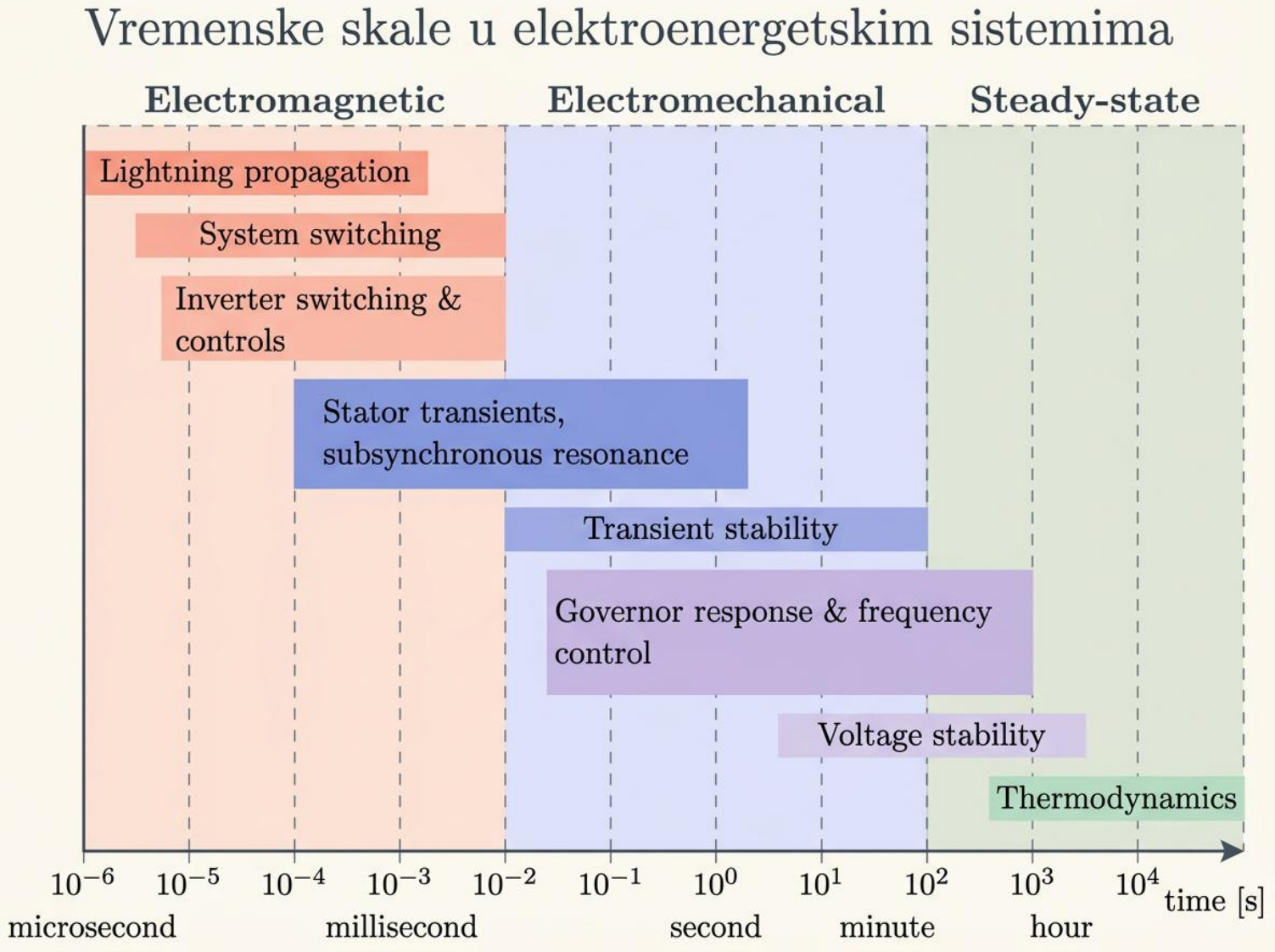
1. Vremenske skale u elektroenergetskim sistemima



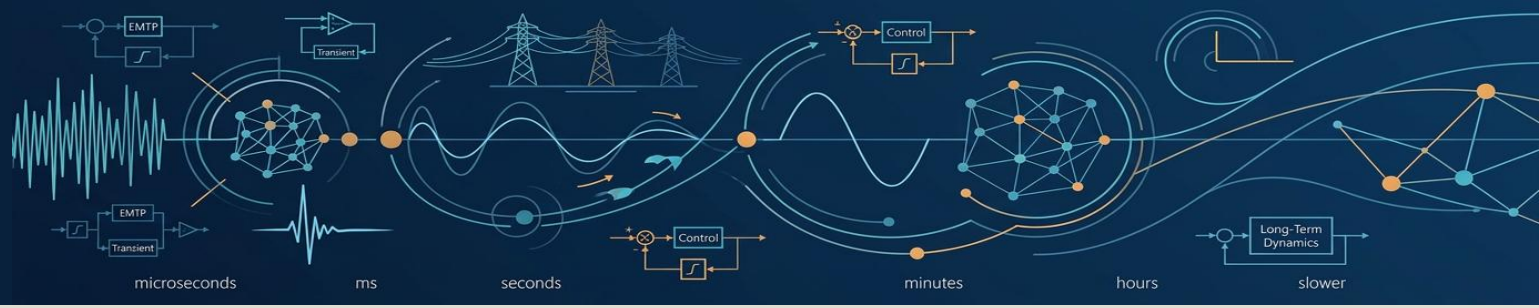
Isti sistem (EES) pokazuje različite fizičke fenomene na različitim vremenskim skalama:

- Kada posmatramo udar groma u dalekovod, relevantne pojave se dešavaju u mikrosekundama
- Kada analiziramo stabilnost nakon kvara, pratimo oscilacije rotora generatora u sekundama

Oba fenomena se dešavaju u istoj mreži, ali zahtevaju potpuno različite modele i pristupe simulaciji.

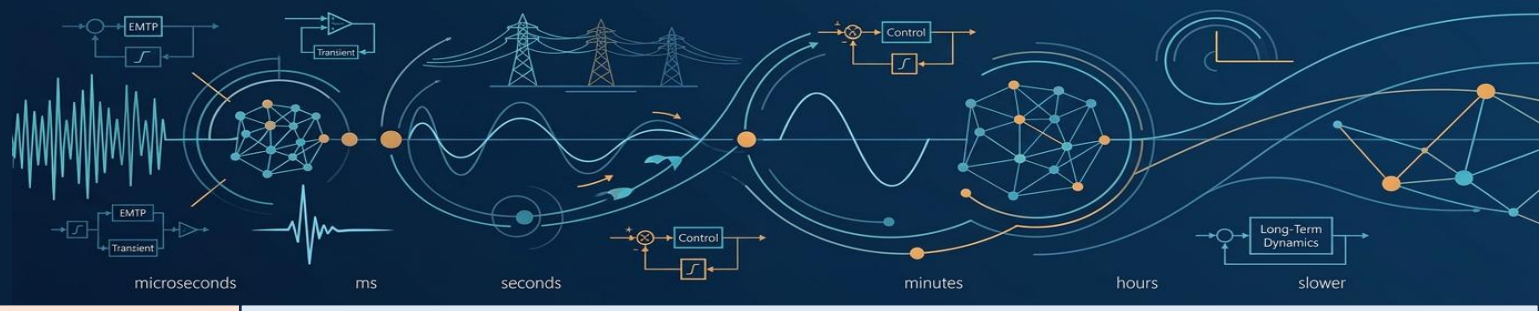


1. Vremenske skale u elektroenergetskim sistemima



Fenomen	Vremenska skala	Primer proračuna	Tip modela
Elektromagnetni	μs – ms	✓ Atmosferski prenaponi ✓ Sklopni prenaponi ✓ Ferorezonansa ✓ Koordinacija izolacije	**EMT** $v(t),\ i(t)$
Elektromehanički	ms – s	✓ Stabilnost ✓ Kratki spojevi	**RMS** $\underline{V}(t),\ \underline{I}(t)$
Kvazi-stacionarni	s – min	✓ Tokovi snaga	Fazorski, iterativni $\underline{V},\ \underline{I}$
Termički procesi	min – h	✓ Zagrevanje transformatora i generatora	Termički modeli $T(t)$

1. Vremenske skale u elektroenergetskim sistemima



Ključne Karakteristike **EMT** Simulacija

- Analiza u vremenskom domenu, rešavanje diferencijalnih jednačina elektroenergetskog sistema u vremenu.
- Precizno i dinamičko ispitivanje ponašanja sistema.
- Detaljne predstave energetske elektronike i upravljačkih sistema.

- Studije koordinacije izolacije
- Ferro-rezonansa
- Studije prekidačkih prenapona
- Studije prenapona usled udara groma

Softver za EMT simulacije

- **PSCAD/EMTDC**
- **EMTP (Electromagnetic Transient Program)**

Ključne Karakteristike **RMS** Simulacija

- Analiza u frekvencijskom domenu, korišćenjem fazorskih proračuna za napone i struje.
- Elektroenergetski sistem je oslobođen brzih tranzijenata i radi u ustaljenom stanju (kvazi-stacionarno).

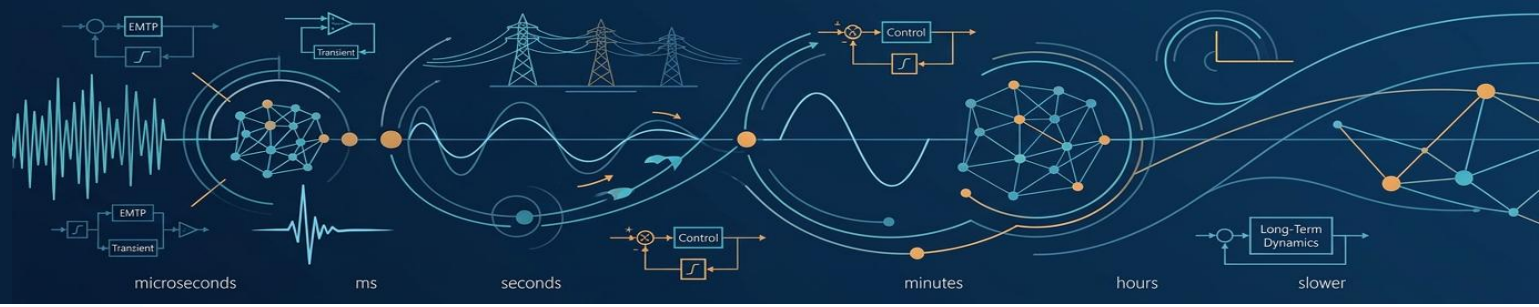
Softver za RMS simulacije

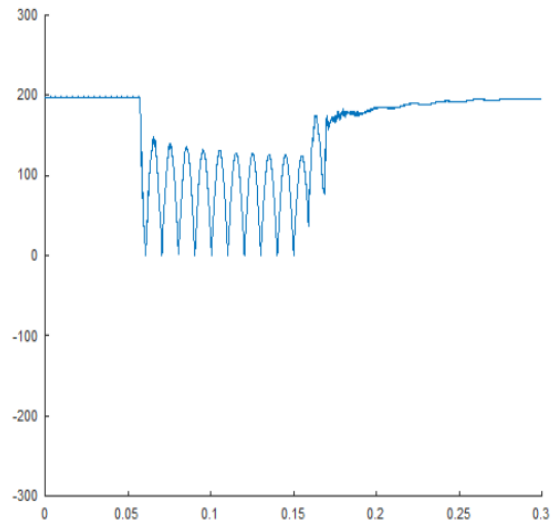
- **DIgSILENT Power Factory**
- **PSS®E (Power System Simulator for Engineering)**

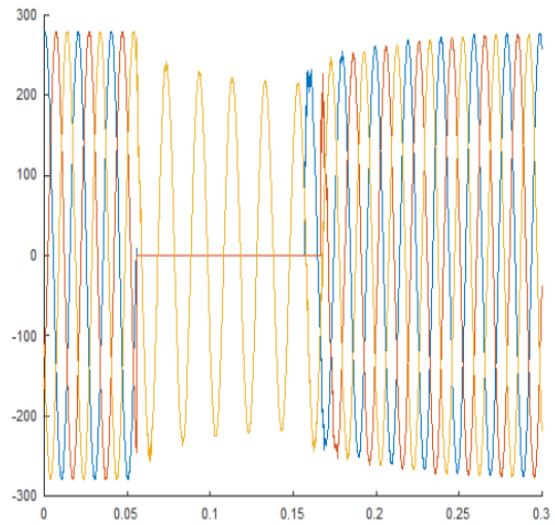
- Studije tokova snaga
- Analiza kratkoročne tranzijentne stabilnosti
- Analiza kratkih spojeva
- Analiza dinamičke stabilnosti

MATLAB® & SIMULINK®

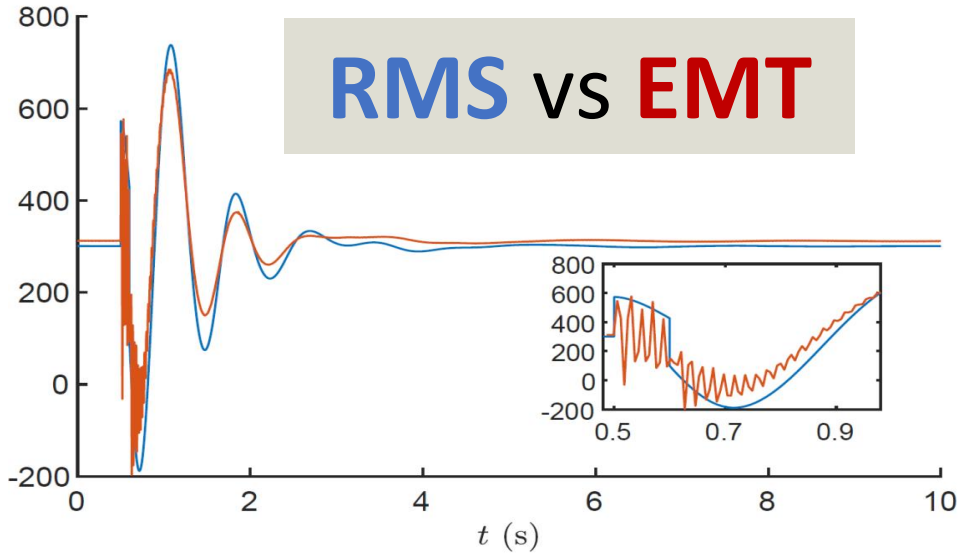
1. Vremenske skale u elektroenergetskim sistemima



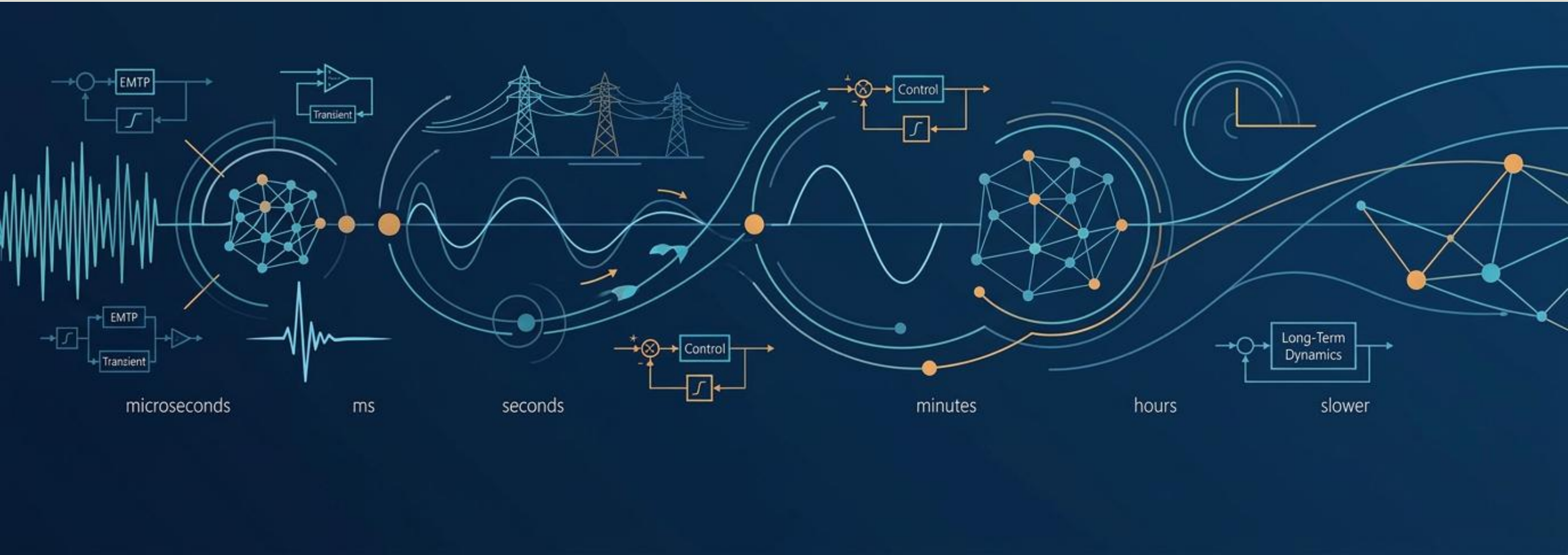




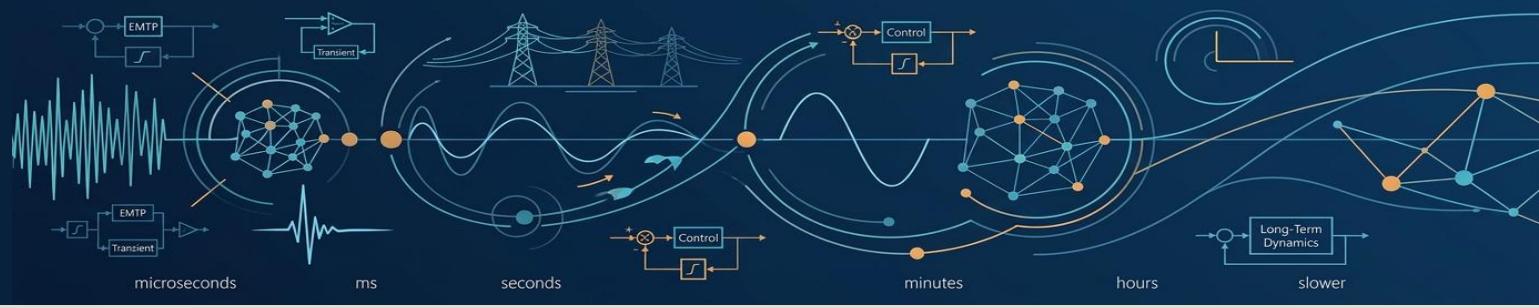
RMS vs **EMT**



2. Tipovi matematičkih jednačina



2. Tipovi matematičkih jednačina



Algebarske jednačine

Definicija: Jednačine koje ne sadrže izvode po vremenu. Promenljive su međusobno povezane, ali nema eksplicitne zavisnosti od vremena.

Opšti oblik:

$$f(x_1, x_2, \dots, x_n) = 0$$

gde su $x_1, x_2, \dots, x_n$ nepoznate veličine.

Jednostavan primer: Ohm-ov zakon u DC kolu:

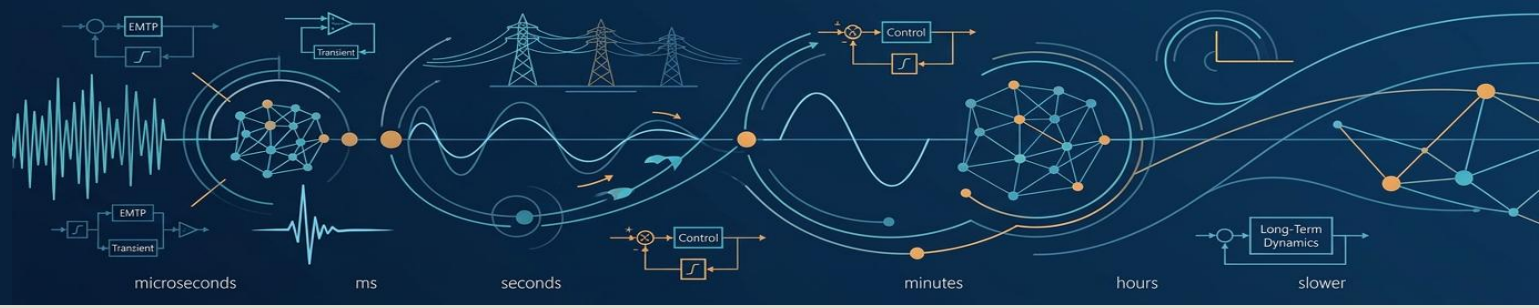
$$V = R \cdot I$$

ili u obliku:

$$V - R \cdot I = 0$$

Ovo je linearna algebarska jednačina. Za kompleksnije sisteme dobijamo sisteme (nelinearnih) algebarskih jednačina.

2. Tipovi matematičkih jednačina



Obične diferencijalne jednačine (ODE – Ordinary Differential Equations)

Definicija: Jednačine koje sadrže **izvode po vremenu** (ili drugoj nezavisnoj promenljivoj), gde sve promenljive zavise samo od **jedne** nezavisne promenljive (obično vreme t).

Opšti oblik ODE prvog reda:

$$\frac{dx}{dt} = f(x,t)$$

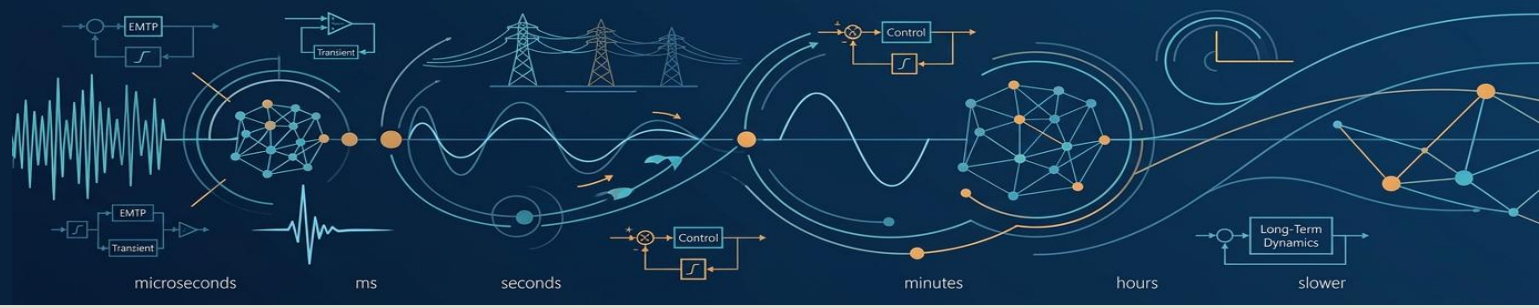
gde su: $x(t)$ – zavisna promenljiva (stanje sistema), t – nezavisna promenljiva (vreme)
$f(x,t)$ – funkcija koja opisuje kako se x menja sa vremenom

Jednostavan primer: RC kolo

$$\frac{dv_C}{dt} = \frac{1}{RC}(V_s - v_C)$$

Ovo je **obična diferencijalna jednačina prvog reda**. Nazivamo je "običnom" jer ima samo jednu nezavisnu promenljivu (t), za razliku od parcijalnih diferencijalnih jednačina koje imaju više nezavisnih promenljivih.

2. Tipovi matematičkih jednačina



Obične diferencijalne jednačine (ODE – Ordinary Differential Equations)

ODE višeg reda:

$$\frac{d^2x}{dt^2} + a\frac{dx}{dt} + bx = 0$$

Ovo je ODE drugog reda (sadrži drugi izvod).

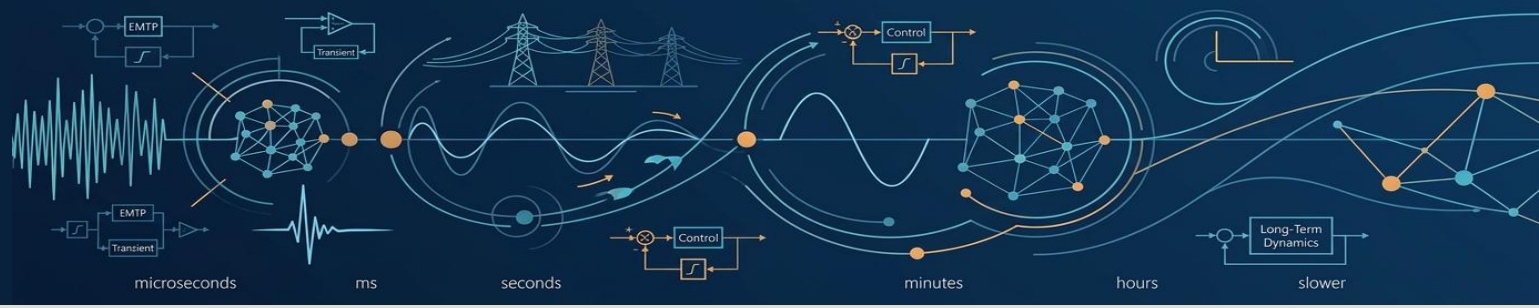
Sistem ODE: U realnim primenama često imamo više promenljivih:

$$\frac{dx_1}{dt} = f_1(x_1, x_2, \dots, x_n, t)$$
$$\frac{dx_2}{dt} = f_2(x_1, x_2, \dots, x_n, t)$$
$$\vdots$$
$$\frac{dx_n}{dt} = f_n(x_1, x_2, \dots, x_n, t)$$

Ovo se zapisuje vektorski:

$$\frac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x}, t)$$

2. Tipovi matematičkih jednačina



Diferencijalno-algebarske jednačine (DAE – Differential-Algebraic Equations)

Definicija: Sistemi jednačina koji sadrže **i diferencijalne i algebarske jednačine**. Neke promenljive imaju dinamiku (izvode), dok su druge definisane algebarskim odnosima.

Opšti oblik DAE:

$$\mathbf{F}\left(\frac{d\mathbf{x}}{dt}, \mathbf{x}, \mathbf{y}, t\right) = 0$$

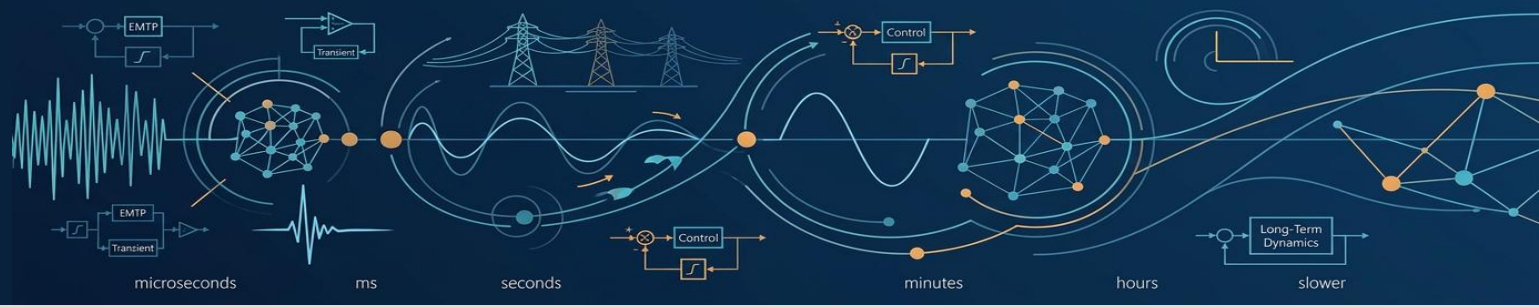
gde su:
$\mathbf{x}(t)$ – **diferencijalne promenljive** (imaju izvode, stanja sistema)
$\mathbf{y}(t)$ – **algebarske promenljive** (bez izvoda, trenutne vrednosti)
Možemo razdvojiti na:
Diferencijalne jednačine:

$$\frac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x}, \mathbf{y}, t)$$

Algebarske jednačine:

$$\mathbf{g}(\mathbf{x}, \mathbf{y}, t) = 0$$

2. Tipovi matematičkih jednačina



Diferencijalno-algebarske jednačine (DAE – Differential-Algebraic Equations)

Jednostavan primer DAE: Generator povezan na mrežu

Dinamika rotora generatora (ODE):

$$M\frac{d\omega}{dt} = P_m - P_e$$

$$\frac{d\delta}{dt} = \omega - \omega_s$$

gde su ω (ugaona brzina) i δ (ugao rotora) **diferencijalne promenljive**.

Električna snaga kroz mrežu (algebarska):

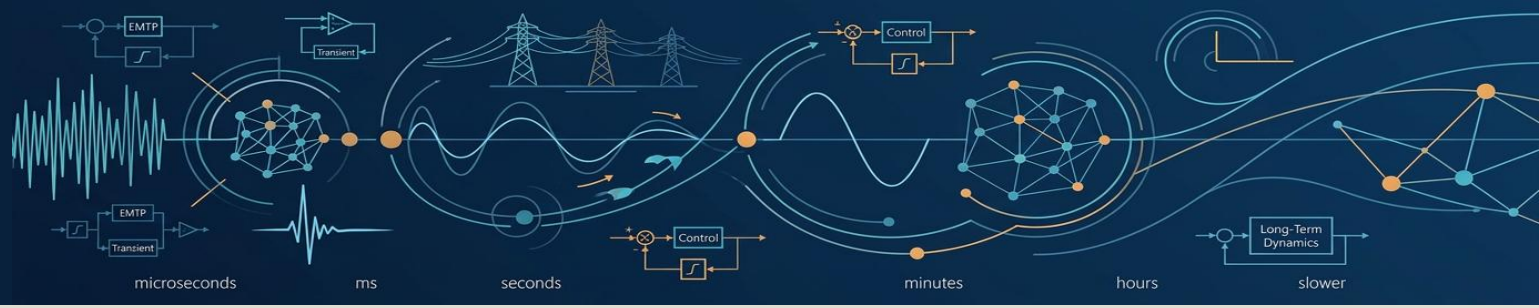
$$P_e = \frac{E \cdot V}{X}\sin(\delta - \theta)$$

$$V\angle\theta = f(\text{stanje mreže, tokovi snaga})$$

Napon V i ugao θ su **algebarske promenljive** – trenutno se prilagođavaju stanju mreže (nema inercije).

Kompletan sistem je **DAE**: kombinacija dinamike rotora (ODE) i trenutnog stanja mreže (algebarske).

2. Tipovi matematičkih jednačina

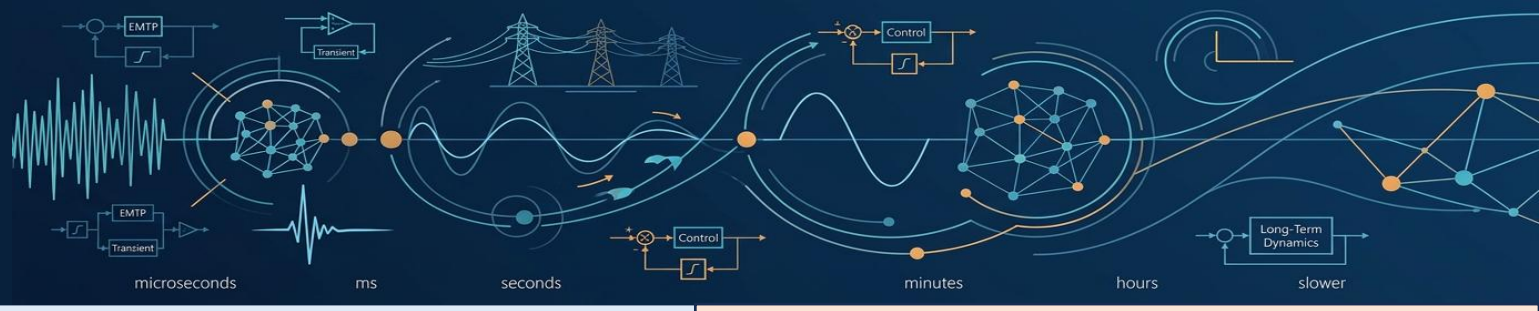


Tip modela	Jednačine	Vremenska konstanta	Tipični korak	Primer
Steady state	Algebarske	–	–	Load flow, kratak spoj
RMS/Phasor	ODE/DAE	ms – s	1–10 ms	Stabilnost, regulatori
EMT	ODE/DAE (detaljne)	μs – ms	1–100 μs	Atmosferski i sklopni prenaponi

Ključna pouka:

- Sve tri kategorije mogu koristiti DAE formu
- Razlika je u **vremenskoj skali**, **detaljnosti** i **koraku simulacije**
- Biramo model prema fenomenu koji nas zanima

2. Tipovi matematičkih jednačina



Steady-state
Nelinearne algebarske jednačine

Primer: Load flow (tokovi snaga)
Balans snage u čvoru i:

$$P_i = \sum_{j=1}^{N} |V_i||V_j||Y_{ij}|\cos(\theta_i - \theta_j - \alpha_{ij})$$

$$Q_i = \sum_{j=1}^{N} |V_i||V_j||Y_{ij}|\sin(\theta_i - \theta_j - \alpha_{ij})$$

Ovo su **nelinearne algebarske jednačine** oblika $f(x) = 0$, gde je x vektor nepoznatih napona i uglova.
Rešavamo ih iterativno (Gauss-Seidel, Newton-Raphson).

RMS dinamika
ODE jednačine

Primer: Obrtna jednačina generatora

$$J\frac{d\omega}{dt} = T_m - T_e - D\omega$$

gde su:

J – moment inercije rotora [kg·m²]
ω – ugaona brzina [rad/s]
T_m – mehanički moment turbine [N·m]
T_e – električni moment [N·m]
D – koeficijent prigušenja

$$M\frac{d^2\delta}{dt^2} = P_m - P_e - D\left(\frac{d\delta}{dt}\right)$$

gde je $M = \frac{2H}{\omega_s}$, a H je konstanta inercije [s].

EMT dinamika
ODE/DAE sa vrlo malim vremenskim korakom

Primer: RLC kolo (prototip EMT modela)

$$\frac{d^2i(t)}{dt^2} + \frac{R}{L}\frac{di(t)}{dt} + \frac{1}{LC}i(t) = \frac{1}{L}\frac{dv_s(t)}{dt}$$

$v_s(t) = 0$ karakteristična jednačina:

$$s^2 + 2\alpha s + \omega_0^2 = 0$$

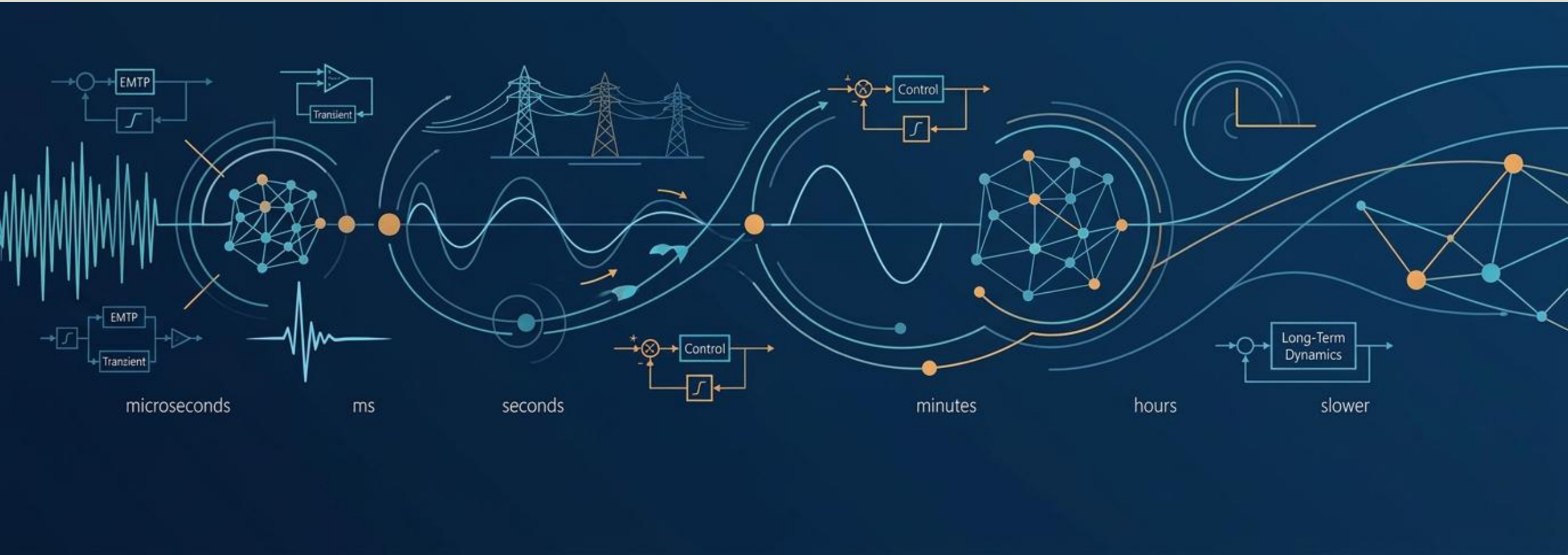
gde su:

$\omega_0 = \frac{1}{\sqrt{LC}}$ – sopstvena frekvencija [rad/s]

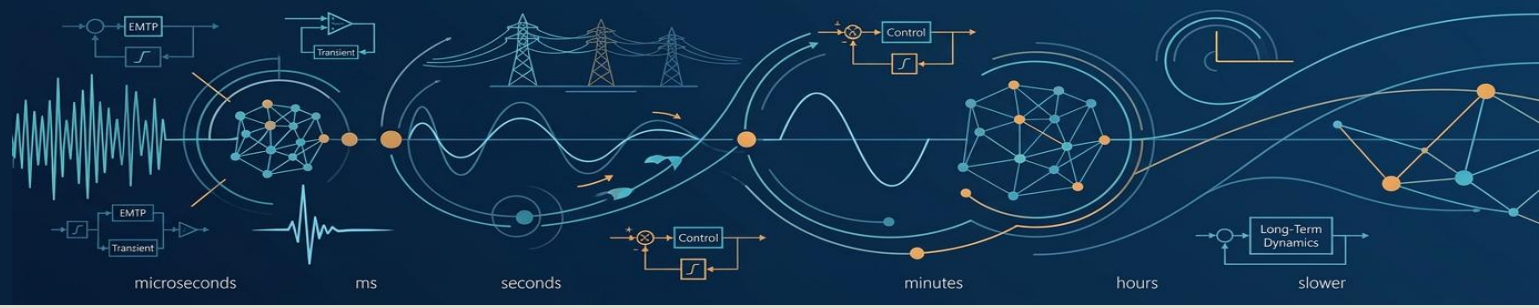
$\alpha = \frac{R}{2L}$ – koeficijent prigušenja [1/s]

$\zeta = \frac{\alpha}{\omega_0} = \frac{R}{2}\sqrt{\frac{C}{L}}$ – faktor prigušenja ($\zeta = 1$)

3. Korak simulacije i računsko vreme



3. Korak simulacije i računsko vreme



Diskretizacija kontinualnog vremena ($dt \rightarrow \Delta t$, $x(t) \rightarrow x_n$, $x(t+\Delta t) \rightarrow x_{n+1}$)

Svaka ODE/DAE mora biti diskretizovana za numeričko rešavanje na računaru. Najčešće metode:

Euler (eksplicitna):

$$x_{n+1} = x_n + \Delta t \cdot f(x_n)$$

Trapezoidna (implicitna):

$$x_{n+1} = x_n + \frac{\Delta t}{2}[f(x_n) + f(x_{n+1})]$$

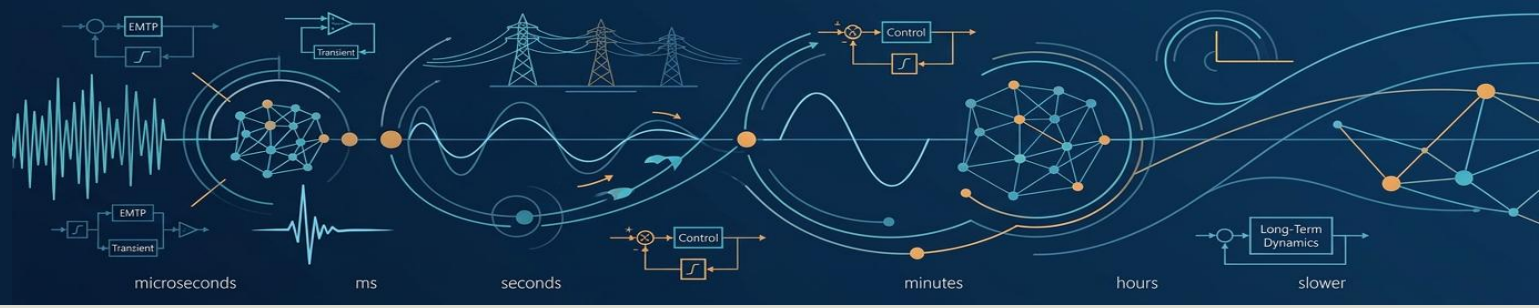
Runge-Kutta 4. reda (RK4):

$$x_{n+1} = x_n + \frac{\Delta t}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

gde su k_1, k_2, k_3, k_4 međurezultati.

$$k_1 = f(t_n, x_n)$$
$$k_2 = f\left(t_n + \frac{\Delta t}{2}, x_n + \frac{\Delta t}{2} k_1\right)$$
$$k_3 = f\left(t_n + \frac{\Delta t}{2}, x_n + \frac{\Delta t}{2} k_2\right)$$
$$k_4 = f(t_n + \Delta t, x_n + \Delta t\, k_3)$$

3. Korak simulacije i računsko vreme



Izbor koraka simulacije Δt

Pravilo: Korak Δt mora biti znatno manji od najmanje vremenske konstante sistema.

$$\Delta t \ll \tau_{\min}$$

Tipično se uzima:

$$\Delta t \approx \frac{\tau_{\min}}{10} \text{ do } \frac{\tau_{\min}}{100}$$

Primer 1: RMS simulacija swing jednačine

$\tau_{\text{swing}} \approx 1$ s

Preporučeni korak: $\Delta t = 1 - 10$ ms

Simulacija 1 s fizičkog vremena sa $\Delta t = 1$ ms: $N = \frac{1}{0.001} = 1{,}000$ koraka

Primer 2: EMT simulacija RC kola

$\tau = RC = 10$ ms

Preporučeni korak: $\Delta t = 10 - 100\ \mu$s

Simulacija 1 s fizičkog vremena sa $\Delta t = 50\ \mu$s: $N = \frac{1}{50\times10^{-6}} = 20{,}000$ koraka

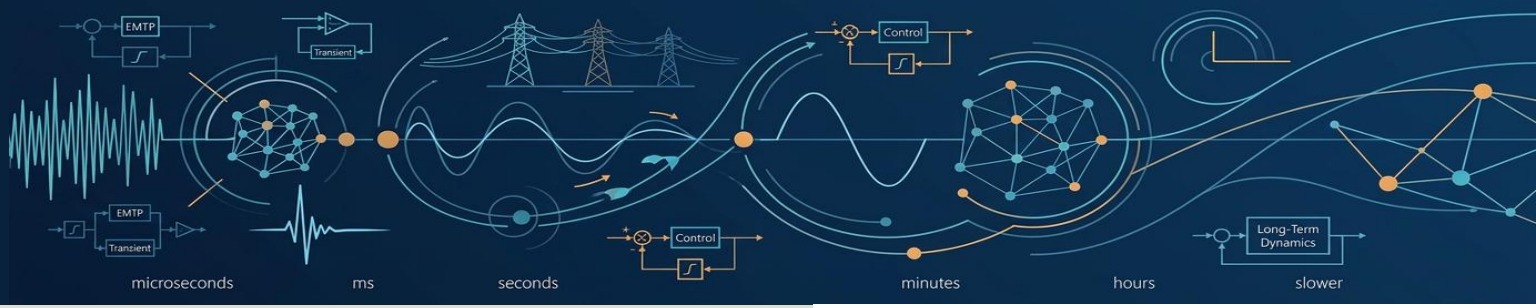
Primer 3: EMT simulacija pretvarača sa PWM na 10 kHz

Perioda PWM-a: $T = 100\ \mu$s

Potrebno je najmanje 10-20 tačaka po periodi → $\Delta t = 5 - 10\ \mu$s

Simulacija 1 s sa $\Delta t = 10\ \mu$s: : $N = \frac{1}{10\times10^{-6}} = 100{,}000$ koraka

3. Korak simulacije i računsko vreme



Računsko vreme i performanse

Računsko vreme zavisi od:

- Broja koraka $N = T_{\text{sim}}/\Delta t$
- Složenosti modela (broj promenljivih, nelinearnosti)
- Tipa solvera (eksplicitni vs. implicitni)

Gruba formula:

$$T_{\text{CPU}} \approx N \times t_{\text{step}}$$

gde je t_{step} vreme CPU-a potrebno za jedan korak simulacije.

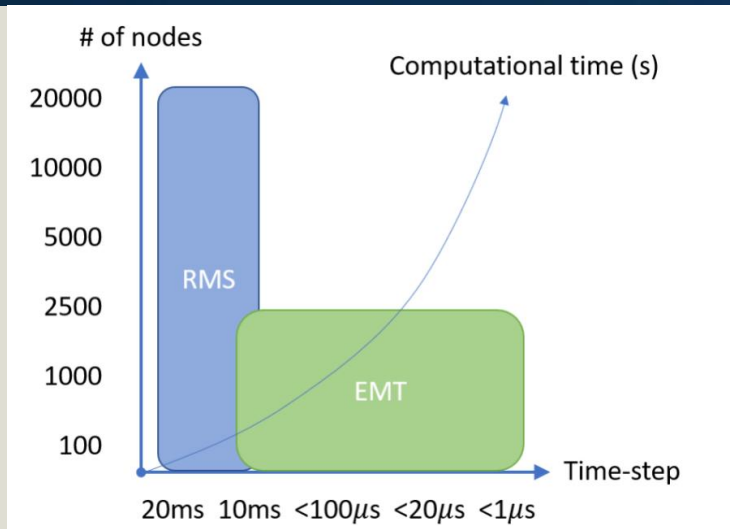
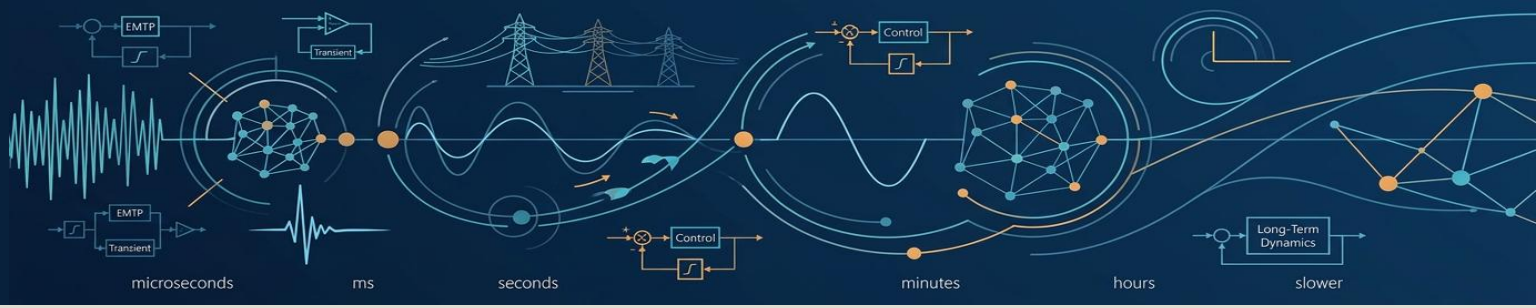


Fig. 4: Number of nodes vs time-step.

Scenario	Δt	T_{sim}	N koraka	T_{CPU} (tipično)
Load flow (Newton-Raphson)	–	–	5–10 iter.	< 1 s
RMS (100-bus, 10 gen.)	1 ms	10 s	10.000	5–30 s
EMT (mali sistem)	50 μs	100 ms	2.000	10–60 s
EMT (veliki sistem)	10 μs	1 s	100.000	minuti–sati

3. Korak simulacije i računsko vreme



Trade-off: Tačnost vs. brzina

Manji korak ($\Delta t \downarrow$):

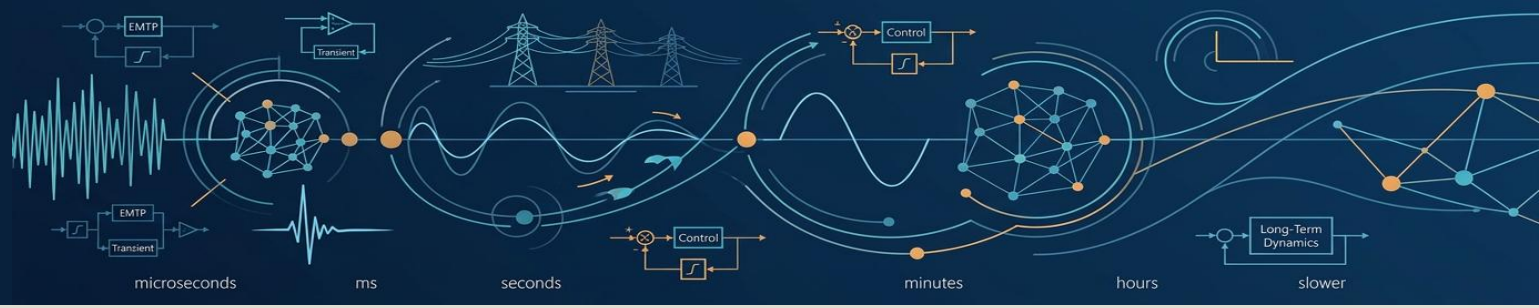
- ✅ Bolja tačnost
- ✅ Stabilniji numerički solver
- ❌ Više računskog vremena
- ❌ Akumulacija greške zaokruživanja

Veći korak ($\Delta t \uparrow$):

- ✅ Brža simulacija
- ❌ Moguća numerička nestabilnost
- ❌ Propuštanje brzih tranzijentnih pojava

Inženjerska odluka: Biramo najmanji korak koji nam daje traženu tačnost uz prihvatljivo računsko vreme.

3. Korak simulacije i računsko vreme



Trade-off: Tačnost vs. brzina

[MATLAB primer RLC kolo]

$$V_s = v_R(t) + v_L(t) + v_C(t)$$

$$\frac{di}{dt} = \frac{V_s - R\,i(t) - v_C(t)}{L}$$

$$\frac{dv_C}{dt} = \frac{i(t)}{C}$$

Euler (eksplicitna) →

$$i_{k+1} = i_k + \Delta t \cdot \left.\frac{di}{dt}\right|_{t_k}$$

$$v_{C,k+1} = v_{C,k} + \Delta t \cdot \left.\frac{dv_C}{dt}\right|_{t_k}$$

Manji korak ($\Delta t \downarrow$):

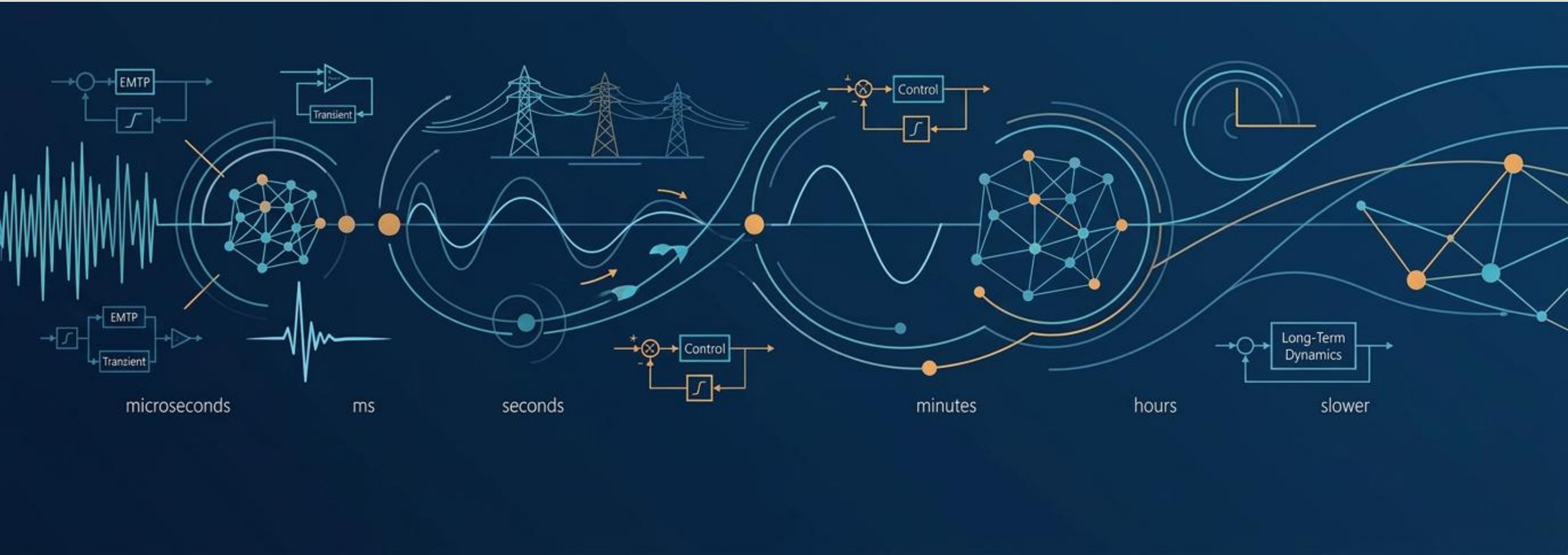
- ✅ Bolja tačnost
- ✅ Stabilniji numerički solver
- ❌ Više računskog vremena
- ❌ Akumulacija greške zaokruživanja

Veći korak ($\Delta t \uparrow$):

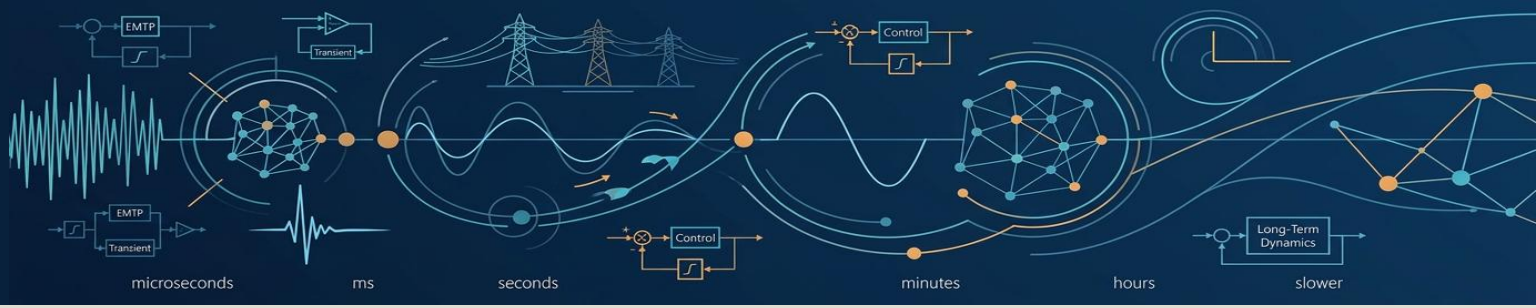
- ✅ Brža simulacija
- ❌ Moguća numerička nestabilnost
- ❌ Propuštanje brzih tranzijentnih pojava

Inženjerska odluka: Biramo najmanji korak koji nam daje traženu tačnost uz prihvatljivo računsko vreme.

4. Koncept real-time simulacije i OPAL-RT



4. Koncept real-time simulacije i OPAL-RT



Offline vs. Real-time simulacija

Offline simulacija:

- Računar rešava model sopstvenom brzinom
- Ako je $\Delta t = 50\ \mu s$ a računaru treba 200 μs po koraku → simulacija traje 4× duže od fizičkog vremena
- **Nema problema** – čekamo dok se završi, dobijamo rezultate

Real-time simulacija:

- Svaki korak mora biti završen tačno za vreme Δt (hard real-time constraint)
- Ako je $\Delta t = 50\ \mu s$ → računar ima samo 50 μs da završi proračun
- Ako ne stigne → **propušta deadline**, simulacija nije validna

Real-time simulacija je postignuta ako:

$$t_{\text{step}} \leq \Delta t$$

gde je:

- t_{step} – vreme CPU-a za jedan korak simulacije
- Δt – korak simulacije (fizičko vreme)

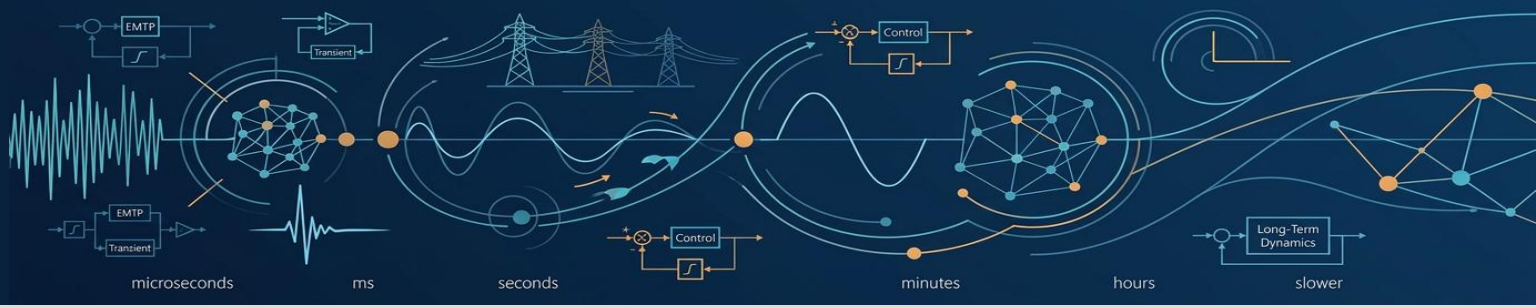
Real-time faktor:

$$\text{RT factor} = \frac{\Delta t}{t_{\text{step}}}$$

- RT factor = 1 → tačno real-time
- RT factor > 1 → brže od realnog vremena (može usporiti)

RT factor < 1 → **ne može real-time**

4. Koncept real-time simulacije i OPAL-RT



Zašto nam treba real-time simulacija?

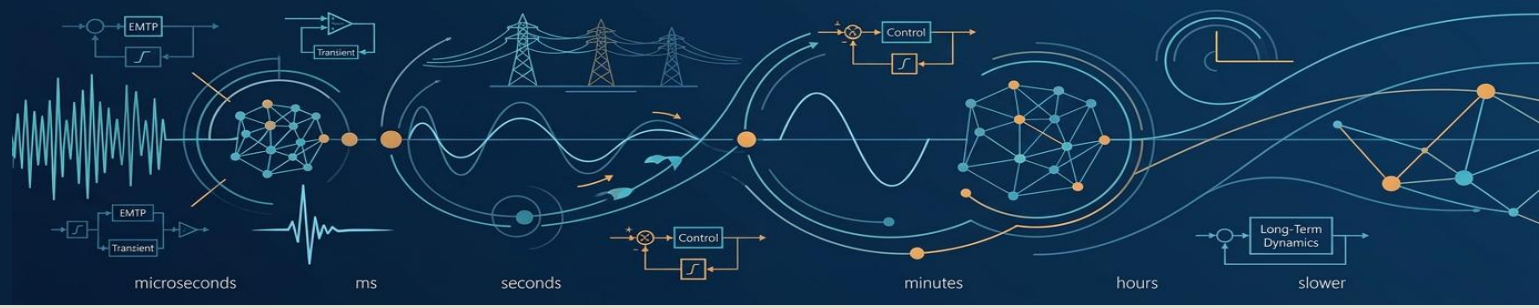
Hardware-in-the-loop (HIL) testiranje:

- Povezivanje realnog hardvera (relej, kontroler, inverter) sa simuliranim sistemom
- Hardver očekuje signale u realnom vremenu → simulator mora da "drži korak"

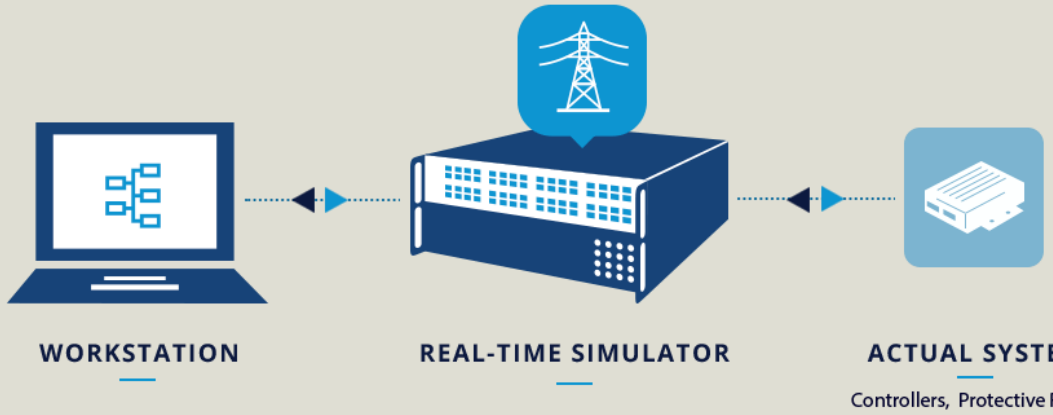
Trening simulatori:

- Obuka – sistem mora reagovati u realnom vremenu na komande

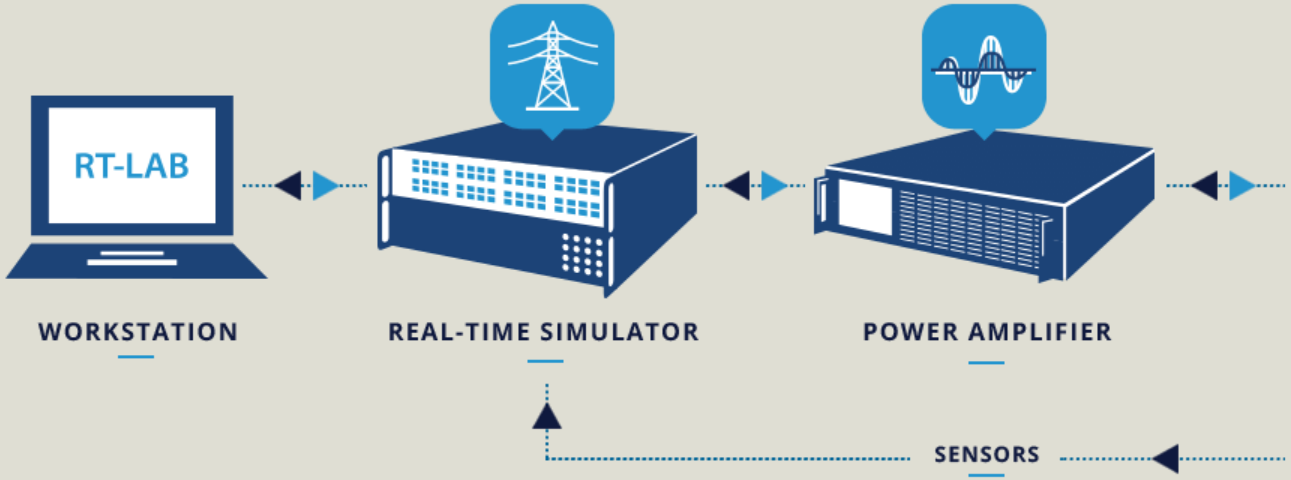
4. Koncept real-time simulacije i OPAL-RT



Zašto nam treba real-time simulacija?

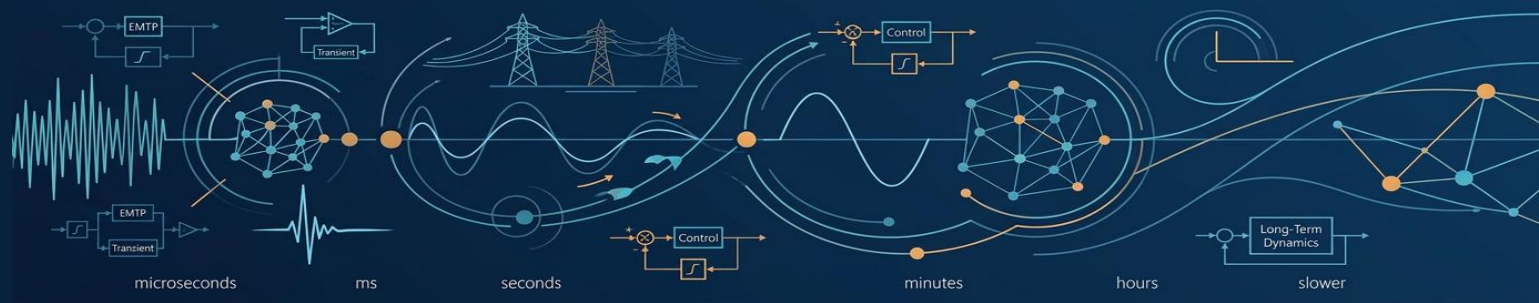


C-HIL



P-HIL

4. Koncept real-time simulacije i OPAL-RT



CPU
(Central Processing Unit)

- Dizajniran da dobro radi sve: operativni sistem, Word, Matlab, browser, …
- Mali broj vrlo snažnih jezgara (npr. 4–16 u modernim procesorima)
- Odličan za raznovrstan kod

Za simulacije EES:

- Dobar za load flow, RMS dinamiku, optimizaciju – manje koraka, kompleksnije operacije po koraku
- Za EMT sa veoma malim korakom (10–50 µs) i velikim brojem čvorova, CPU često ne stiže da završi sve u zadatom real-time intervalu

GPU
(Graphics Processing Unit)

- Sadrži stotine ili hiljade jednostavnih jezgara koja rade isti tip operacija nad velikim vektorima/poljima podataka
- Odličan za: množenje matrica, vektorske operacije, Monte Carlo simulacije, deep learning
- Svako jezgro je slabije od CPU jezgra, ali ih ima mnogo

Za simulacije EES:

- Koristan za ubrzanje velikih matrica i linearne algebre (npr. faktorizacija matrice admitanse ili sistema jednačina u EMT)
- Tipičan obrazac: CPU upravlja simulacijom, a GPU rešava "tešku" numeriku (npr. sistem linearnih jednačina u svakom koraku)

Matlab primena: https://www.mathworks.com/videos/introduction-to-gpu-computing-with-matlab-1615529245694.html

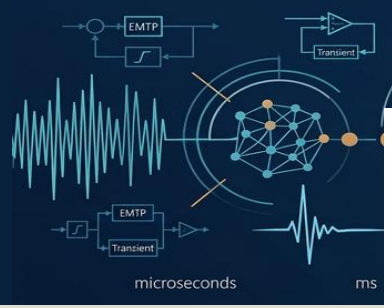
FPGA
(Field-Programmable Gate Array)

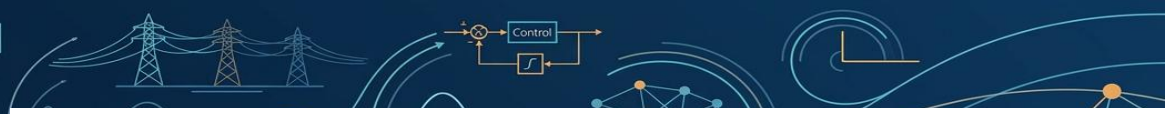
- FPGA se ponaša kao specijalno proizveden **čip samo za određen algoritam** (npr. za rešavanje EMT jednačina)
- Ogroman stepen paralelizma: stotine ili hiljade operacija se odvijaju istovremeno
- **Deterministička kašnjenja**: zna se Koliko će svaka operacija trajati

Za real-time simulaciju EES:

- Vrlo mali korak (npr. 1 µs ili manje)
- Veliki broj istovremenih operacija (više grana mreže, više poluvodičkih prekidača)
- Strogo determinisano vreme koraka (bez "skokova" kao na PC-ju)

4. Koncept real-time simulacije i OPAL-RT





Osnovni koncept: CPU ↔ GPU transfer podataka

Osnovna šema rada sa GPU u Matlabu:

1. **Kreiraj podatke na CPU** (obična Matlab promenljiva)
2. **Prebaci na GPU** sa gpuArray()
3. **Radi računanje na GPU** (Matlab automatski koristi GPU verzije funkcija)
4. **Vrati rezultate na CPU** sa gather()

```
% Provera da li GPU postoji i može se koristiti
try
gpu = gpuDevice;
fprintf('GPU pronađen: %s\n', gpu.Name);
fprintf('Compute capability: %.1f\n', gpu.ComputeCapability);
fprintf('Memorija: %.2f GB\n', gpu.TotalMemory/1e9);
fprintf('Multiprocessors: %d\n', gpu.MultiprocessorCount);
catch
fprintf('GPU nije dostupan ili Parallel Computing Toolbox nije
instaliran.\n');
end
```

```
% 1. Kreiranje matrica na CPU
N = 5000;
A = rand(N, N);
B = rand(N, N);
% 2. Množenje na CPU
tic;
C_cpu = A * B;
t_cpu = toc;
fprintf('CPU vreme: %.4f s\n', t_cpu);
% 3. Prebacivanje na GPU
tic;
A_gpu = gpuArray(A);
B_gpu = gpuArray(B);
t_transfer = toc;
% 4. Množenje na GPU
tic;
C_gpu = A_gpu * B_gpu;
wait(gpuDevice); % Čekaj da GPU završi
t_gpu = toc;
% 5. Vraćanje rezultata na CPU
tic;
C_result = gather(C_gpu);
t_gather = toc;
fprintf('GPU transfer vreme: %.4f s\n', t_transfer);
fprintf('GPU računanje: %.4f s\n', t_gpu);
fprintf('GPU→CPU transfer: %.4f s\n', t_gather);
fprintf('Ukupno GPU: %.4f s\n', t_transfer + t_gpu + t_gather);
fprintf('Ubrzanje: %.2fx\n', t_cpu / (t_gpu));
```

Hvala na pažnji!