Recent Developments in the Dynamic Optimisation Package DYNO

F. LESAGE, M. FIKAR*, M.A. LATIFI

Laboratoire des Sciences du Génie Chimique

1, rue Grandville

54000 NANCY – FRANCE

* Department of Information Engineering and Process Control

FCHPT STU

Radlinského 9

812 37 BRATISLAVA – SLOVAKIA

Dynamic optimisation

Find $\mathbf{u}(t)$ and/or $\mathbf{p}$ so that a cost function J_0 is minimal:

$$\min_{\mathbf{u},\mathbf{p}} J_0 = G_0(\mathbf{x}(t_f), \mathbf{p}, t_f) + \int_{t_0}^{t_f} F_0(\mathbf{x}, \mathbf{u}, \mathbf{p}, t)\, dt$$

Dynamic optimisation

Find $\mathbf{u}(t)$ and/or $\mathbf{p}$ so that a cost function J_0 is minimal:

$$\min_{\mathbf{u},\mathbf{p}} J_0 = G_0(\mathbf{x}(t_f), \mathbf{p}, t_f) + \int_{t_0}^{t_f} F_0(\mathbf{x}, \mathbf{u}, \mathbf{p}, t)\, dt$$

Dynamic optimisation

Find $\mathbf{u}(t)$ and/or $\mathbf{p}$ so that a cost function J_0 is minimal:

$$\min_{\mathbf{u},\mathbf{p}} J_0 = G_0(\mathbf{x}(t_f), \mathbf{p}, t_f) + \int_{t_0}^{t_f} F_0(\mathbf{x}, \mathbf{u}, \mathbf{p}, t)\, dt$$

Dynamic optimisation

Find $\mathbf{u}(t)$ and/or $\mathbf{p}$ so that a cost function J_0 is minimal:

$$\min_{\mathbf{u},\mathbf{p}} J_0 = G_0(\mathbf{x}(t_f), \mathbf{p}, t_f) + \int_{t_0}^{t_f} F_0(\mathbf{x}, \mathbf{u}, \mathbf{p}, t)\, dt$$

subject to:

$$\begin{aligned}
\dot{\mathbf{x}} &= \mathbf{f}(\mathbf{x}, \mathbf{u}, \mathbf{p}, t) \\
\mathbf{x}(0) &= \mathbf{x}_0(\mathbf{p}) \\
J_i &= 0 \; i \in \{1..m_e\} \\
J_i &\geq 0 \; i \in \{m_e + 1..m\} \\
\mathbf{x}_L &\leq \mathbf{x} \leq \mathbf{x}_U \\
\mathbf{u}_L &\leq \mathbf{u} \leq \mathbf{u}_U \\
\mathbf{p}_L &\leq \mathbf{p} \leq \mathbf{p}_U
\end{aligned}$$

Canonical form

Constraints J_i ($1 \leq i \leq m$) can be expressed like performance index J_0:

$$J_i = G_i(\mathbf{x}(t_f), \mathbf{p}, t_f) + \int_0^{t_f} F_i(\mathbf{x}, \mathbf{u}, \mathbf{p}, t)\, dt \; 1 \leq i \leq m$$

☞ Same form for all J_i ($0 \leq i \leq m$)

Control parametrisation

- Time discretisation:
  N intervals of length Δt_j ($1 \leq j \leq N$)

Control parametrisation

- Time discretisation:
  N intervals of length Δt_j ($1 \leq j \leq N$)
- Control Vector Parametrization (piece-wise constant):
  $\mathbf{u}(t) = \mathbf{u}_j \;\; t_{j-1} \leq t \leq t_j$

Control parametrisation

- Time discretisation:
  N intervals of length Δt_j ($1 \leq j \leq N$)
- Control Vector Parametrization (piece-wise constant):
  $\mathbf{u}(t) = \mathbf{u}_j \;\; t_{j-1} \leq t \leq t_j$

$\rightarrow$ new set of optimised variables:
$\mathbf{y}^T = (\Delta t_1, \ldots, \Delta t_P, \mathbf{u}_1^T, \ldots, \mathbf{u}_P^T, \mathbf{p}^T)$

Control parametrisation

- Time discretisation:
  N intervals of length Δt_j ($1 \leq j \leq N$)
- Control Vector Parametrization (piece-wise constant):
  $\mathbf{u}(t) = \mathbf{u}_j \;\; t_{j-1} \leq t \leq t_j$

$\rightarrow$ new set of optimised variables:
$\mathbf{y}^T = (\Delta t_1, \ldots, \Delta t_P, \mathbf{u}_1^T, \ldots, \mathbf{u}_P^T, \mathbf{p}^T)$
Model is included in augmented cost/constraints functions:

$$\bar{J}_i = G_i + \int_{t_0}^{t_P} \left(F_i + \boldsymbol{\lambda}_i^T(t) \cdot (\mathbf{f} - \dot{\mathbf{x}}) \right) dt \; 0 \leq i \leq m$$

Control parametrisation

- Time discretisation:
  N intervals of length Δt_j ($1 \leq j \leq N$)
- Control Vector Parametrization (piece-wise constant):
  $\mathbf{u}(t) = \mathbf{u}_j \;\; t_{j-1} \leq t \leq t_j$

$\rightarrow$ new set of optimised variables:
$\mathbf{y}^T = (\Delta t_1, \ldots, \Delta t_P, \mathbf{u}_1^T, \ldots, \mathbf{u}_P^T, \mathbf{p}^T)$
Model is included in augmented cost/constraints functions:

$$\bar{J}_i = G_i + \int_{t_0}^{t_P} \Big(F_i + \boldsymbol{\lambda}_i^T(t) \cdot (\mathbf{f} - \dot{\mathbf{x}}) \Big) \, dt \; 0 \leq i \leq m$$

☞ Hamiltonian $H_i = -F_i + \boldsymbol{\lambda}_i^T \cdot \mathbf{f} \; 0 \leq i \leq m$

Resulting problem

Leading to a non-linear programming optimisation problem

$$\min_{\mathbf{y}} J_0(\mathbf{y})$$

subject to:

$$\begin{aligned} \bar{J}_i(\mathbf{y}) &= 0 \qquad & i \in \{1..m_e\} \\ \bar{J}_i(\mathbf{y}) &\geq 0 \qquad & i \in \{m_e + 1..m\} \end{aligned}$$

☞ SQP solver (NLPQL, SLSQP)

☞ gradients of $\bar{J}_0$ and $\bar{J}_i$ with respect to $\mathbf{y}$ must be known

- finite differences
- user-supplied functions (hand-made, ADIFOR,...)

Gradients computation

Variations of augmented performance index and constraints:

$$\delta \bar{J}_i = \left[H_i(t_P^-) + \frac{\partial G_i}{\partial t_P} \right] \delta t_P + \sum_{j=1}^{P-1} \left[H_i(t_j^-) - H_i(t_j^+) + \frac{\partial G_i}{\partial t_j} \right] \delta t_j +$$

$$\sum_{j=1}^{P} \left[\frac{\partial G_i}{\partial \mathbf{u}^T} + \int_{t_{j-1}^+}^{t_j^-} \frac{\partial H_i}{\partial \mathbf{u}^T} \, dt \right] \delta \mathbf{u}_j + \left[\boldsymbol{\lambda}^T(t_0^+) \frac{\partial \mathbf{x}_0}{\partial \mathbf{p}^T} + \int_{t_0}^{t_P} \frac{\partial H_i}{\partial \mathbf{p}} \, dt \right] \delta p$$

$(0 \leq i \leq m)$

Gradients computation

Variations of augmented performance index and constraints:

$$\delta \bar{J}_i = \left[H_i(t_P^-) + \frac{\partial G_i}{\partial t_P} \right] \delta t_P + \sum_{j=1}^{P-1} \left[H_i(t_j^-) - H_i(t_j^+) + \frac{\partial G_i}{\partial t_j} \right] \delta t_j +$$

$$\sum_{j=1}^{P} \left[\frac{\partial G_i}{\partial \mathbf{u}^T} + \int_{t_{j-1}^+}^{t_j^-} \frac{\partial H_i}{\partial \mathbf{u}^T} \, dt \right] \delta \mathbf{u}_j + \left[\boldsymbol{\lambda}^T(t_0^+) \frac{\partial \mathbf{x}_0}{\partial \mathbf{p}^T} + \int_{t_0}^{t_P} \frac{\partial H_i}{\partial \mathbf{p}} \, dt \right] \delta p$$

$(0 \leq i \leq m)$

Gradients computation

Variations of augmented performance index and constraints:

$$\delta \bar{J}_i = \left[H_i(t_P^-) + \frac{\partial G_i}{\partial t_P} \right] \delta t_P + \sum_{j=1}^{P-1} \left[H_i(t_j^-) - H_i(t_j^+) + \frac{\partial G_i}{\partial t_j} \right] \delta t_j +$$

$$\sum_{j=1}^{P} \left[\frac{\partial G_i}{\partial \mathbf{u}^T} + \int_{t_{j-1}^+}^{t_j^-} \frac{\partial H_i}{\partial \mathbf{u}^T} \, dt \right] \delta \mathbf{u}_j + \left[\boldsymbol{\lambda}^T(t_0^+) \frac{\partial \mathbf{x}_0}{\partial \mathbf{p}^T} + \int_{t_0}^{t_P} \frac{\partial H_i}{\partial \mathbf{p}} \, dt \right] \delta p$$

$(0 \leq i \leq m)$

☞ Terms in red are gradients with respect to t_P, t_j, $\mathbf{p}$ and u_j (used by NLP solver)

☞ Gradients with respect to $\mathbf{x}$ are nullified by a proper choice of λ (optimality condition)

Gradients computation (2)

- Forward integration of $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, \mathbf{p}, t)$ and computation of integrand terms F_i;

Gradients computation (2)

- Forward integration of $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, \mathbf{p}, t)$ and computation of integrand terms F_i;
- Computation of $\boldsymbol{\lambda}_i(t)$.

Adjoint system

We choose $\boldsymbol{\lambda}$ so that:

$$\dot{\boldsymbol{\lambda}}_i^T(t) = -\frac{\partial H_i}{\partial \mathbf{x}^T}$$

with terminal conditions for each time interval:

$$\begin{aligned}
\boldsymbol{\lambda}_i^T(t_P) &= \frac{\partial G_i}{\partial \mathbf{x}^T}(t_P) \\
\boldsymbol{\lambda}_i^T(t_j^-) &= \boldsymbol{\lambda}_i^T(t_j^+) + \frac{\partial G_i}{\partial \mathbf{x}^T}(t_j)
\end{aligned}$$

☞ Backward integration of adjoint system
☞ Need to know states $\mathbf{x}$: $\mathbf{x}$ are stored at discrete time units when computed and interpolated during backward integration

Gradients computation (2)

- Forward integration of $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, \mathbf{p}, t)$ and computation of integrand terms F_i ...

Gradients computation (2)

- Forward integration of $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, \mathbf{p}, t)$ and computation of integrand terms F_i ...
- Backward integration of $\dot{\boldsymbol{\lambda}}_i = -\frac{\partial H_i}{\partial \mathbf{x}^T}$, computation of

$\int_{t_{j-1}^+}^{t_j^-} \frac{\partial H_i}{\partial \mathbf{u}^T}\, dt$ and $\int_{t_{j-1}^+}^{t_j^-} \frac{\partial H_i}{\partial \mathbf{p}^T}\, dt$

Gradients computation (2)

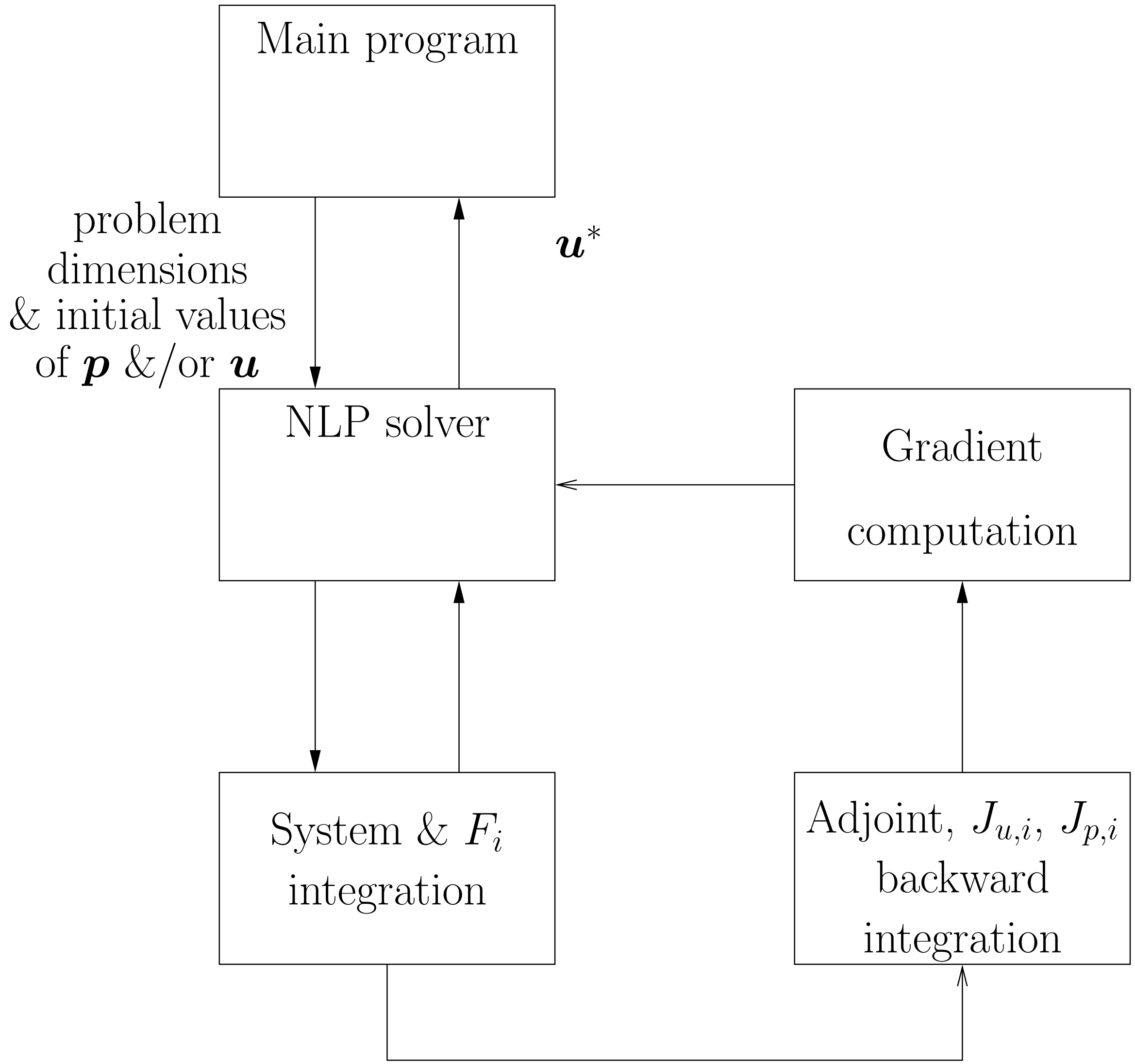
- Forward integration of $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, \mathbf{p}, t)$ and computation of integrand terms F_i ...
- Backward integration of $\dot{\boldsymbol{\lambda}}_i = -\frac{\partial H_i}{\partial \mathbf{x}^T}$, computation of $\int_{t_{j-1}^+}^{t_j^-} \frac{\partial H_i}{\partial \mathbf{u}^T}\, dt$ and $\int_{t_{j-1}^+}^{t_j^-} \frac{\partial H_i}{\partial \mathbf{p}^T}\, dt$
- Computation of other gradients $\rightarrow$ gradients of all $\bar{J}_i$ ($0 \leq i \leq m$)

Gradients computation (2)

- Forward integration of $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, \mathbf{p}, t)$ and computation of integrand terms F_i ...
- Backward integration of $\dot{\boldsymbol{\lambda}}_i = -\frac{\partial H_i}{\partial \mathbf{x}^T}$, computation of $\int_{t_{j-1}^+}^{t_j^-} \frac{\partial H_i}{\partial \mathbf{u}^T}\, dt$ and $\int_{t_{j-1}^+}^{t_j^-} \frac{\partial H_i}{\partial \mathbf{p}^T}\, dt$
- Computation of other gradients $\rightarrow$ gradients of all $\bar{J}_i$ $(0 \leq i \leq m)$

☞ All gradients and values are sent back to NLP solver

Overview of sequence



Example

$$\min_{u} J_0 = \int_0^1 (x^2 + u^2)\, dt$$

with:

$$\begin{aligned} \dot{x}(t) &= u(t) \\ x(0) &= 1 \\ J_1 = x(1) &= 0.5 \\ J_2 = x(0.6) &= 0.8 \end{aligned}$$

User's choise: N=10 time intervals
Dyno's formulation:
$G_0 = 0$, $F_0 = x^2 + u^2$
$G_1 = x(t_{10}) - 0.5$, $F_1 = 0$ and $G_2 = x(t_6) - 0.8$, $F_1 = 0$

Example (2)

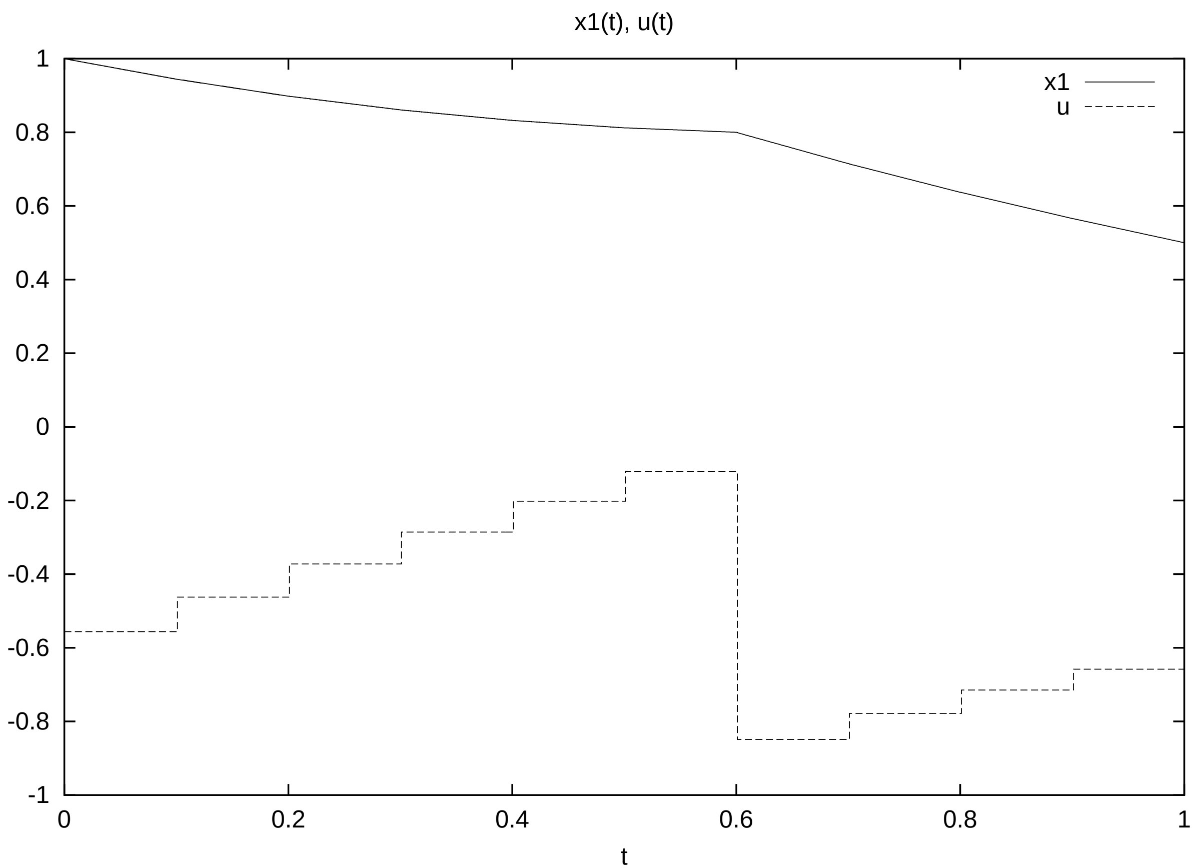


Figure 1: Results for test problem: state and optimal control variable

Example (2)

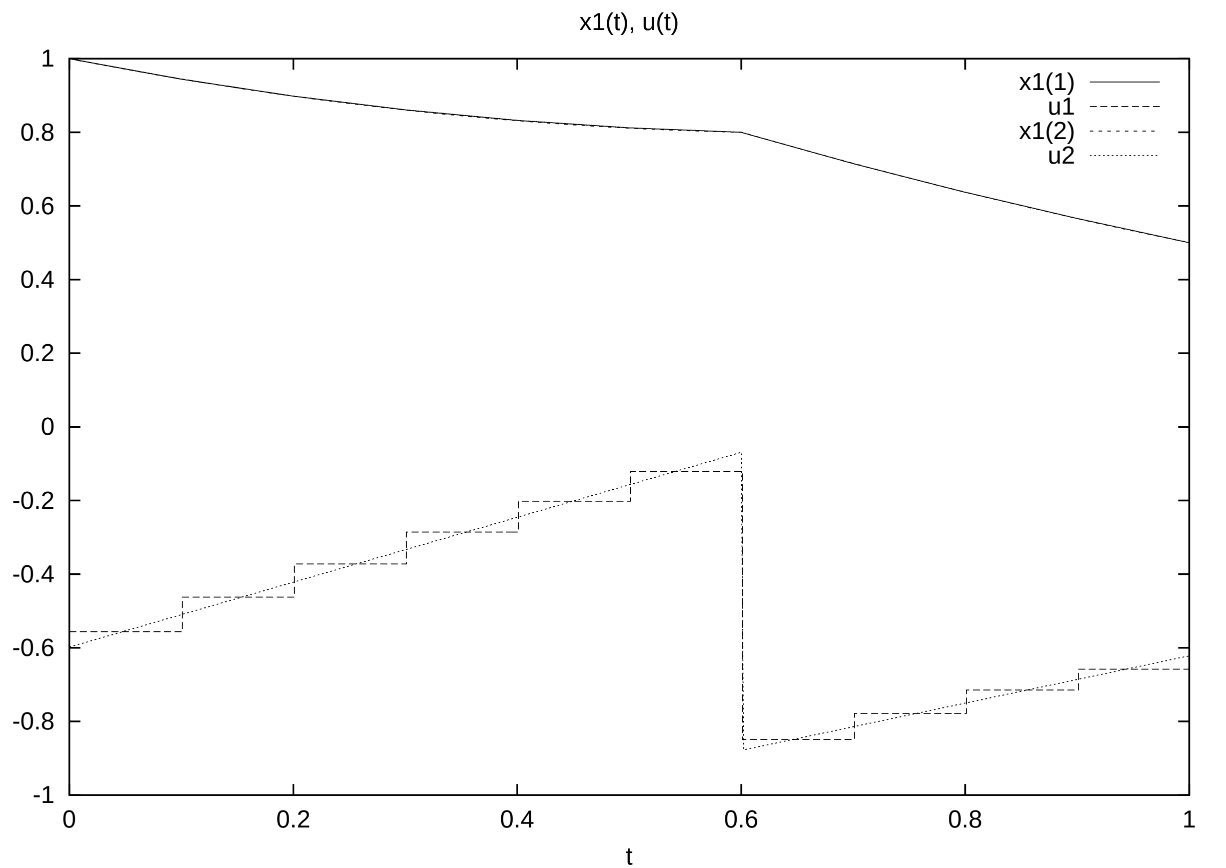
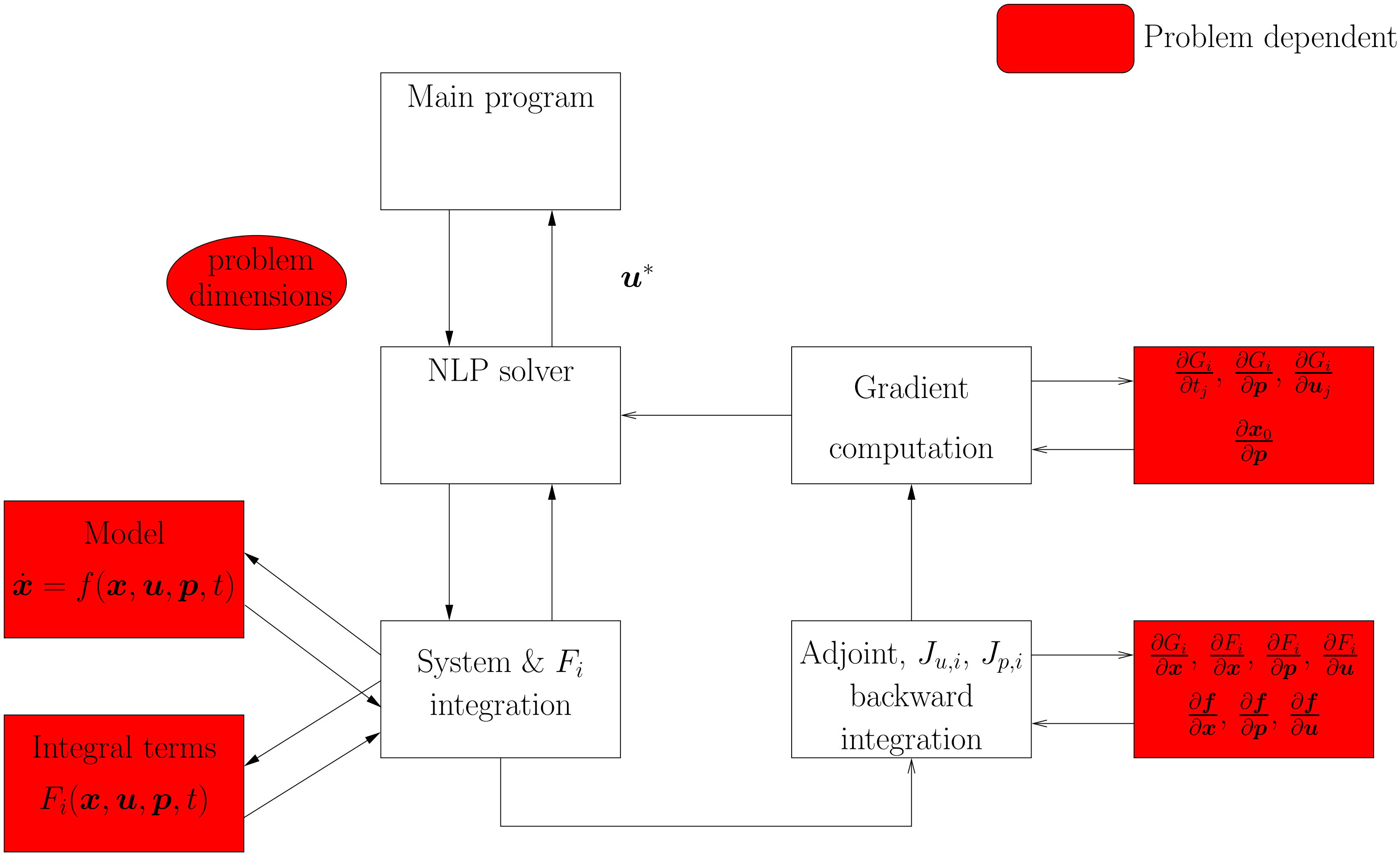
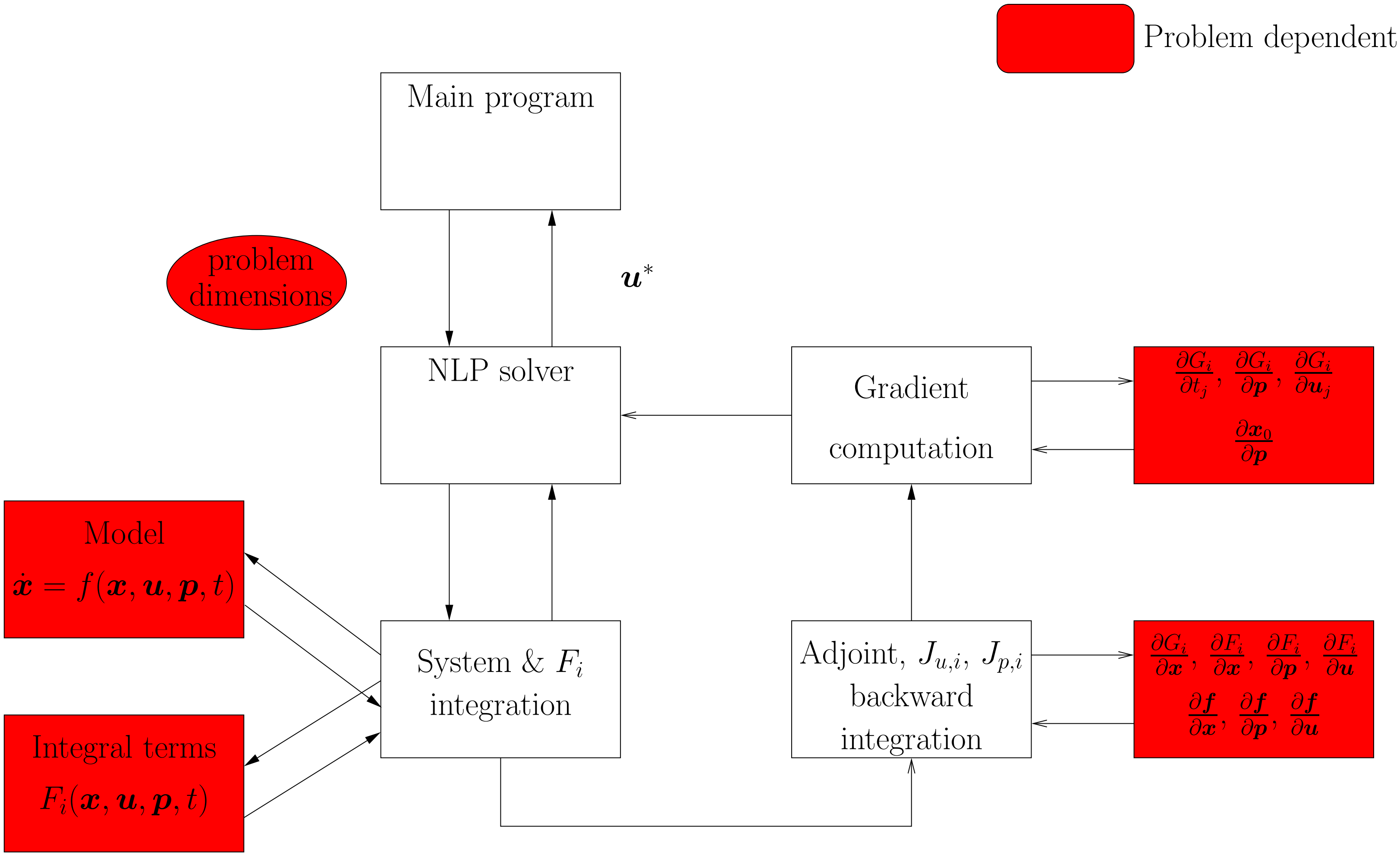


Figure 1: Comparison of optimal trajectories for $N = 10$ and piece-wise linear control & $N = 2$

Matlab integration

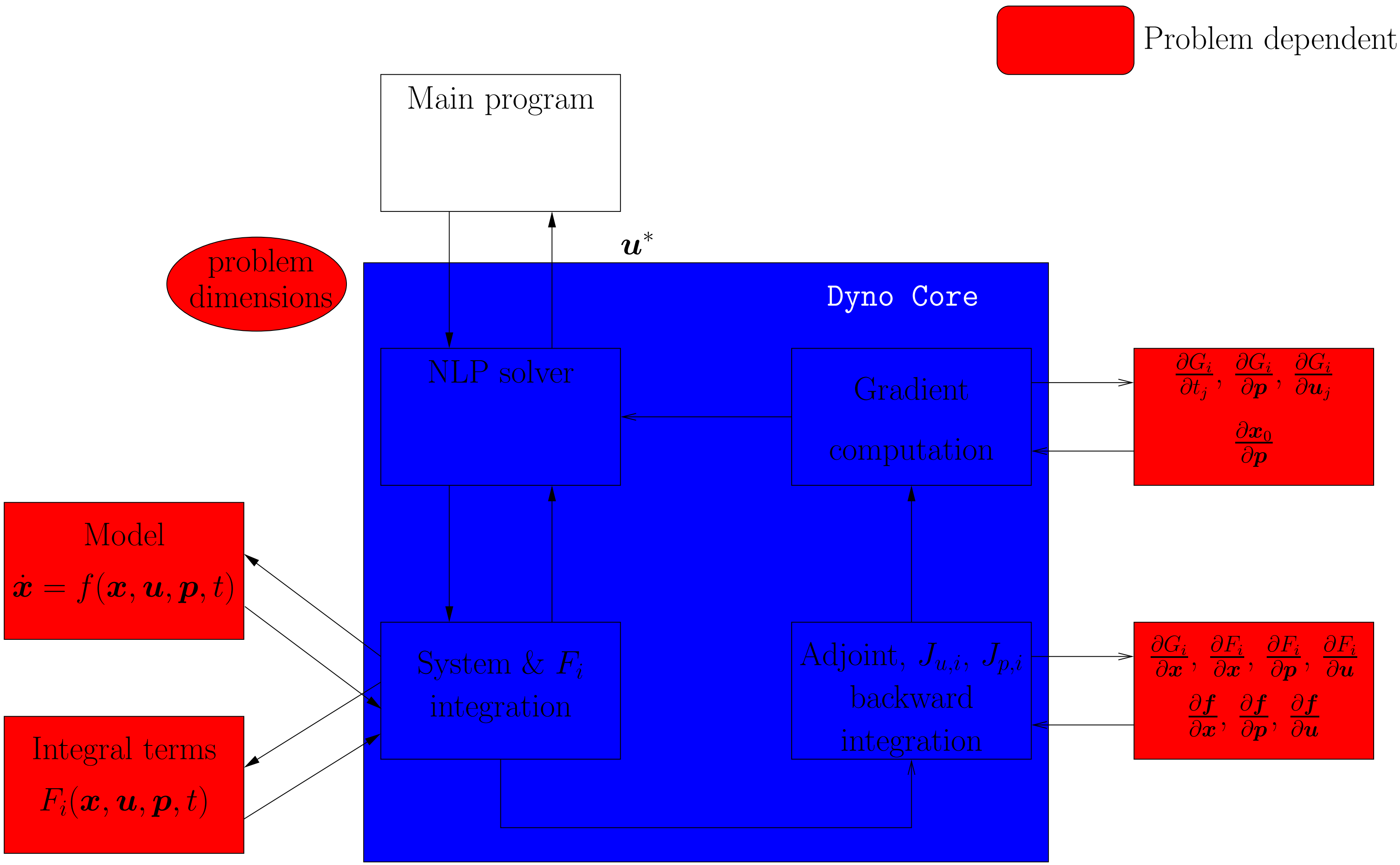


Matlab integration

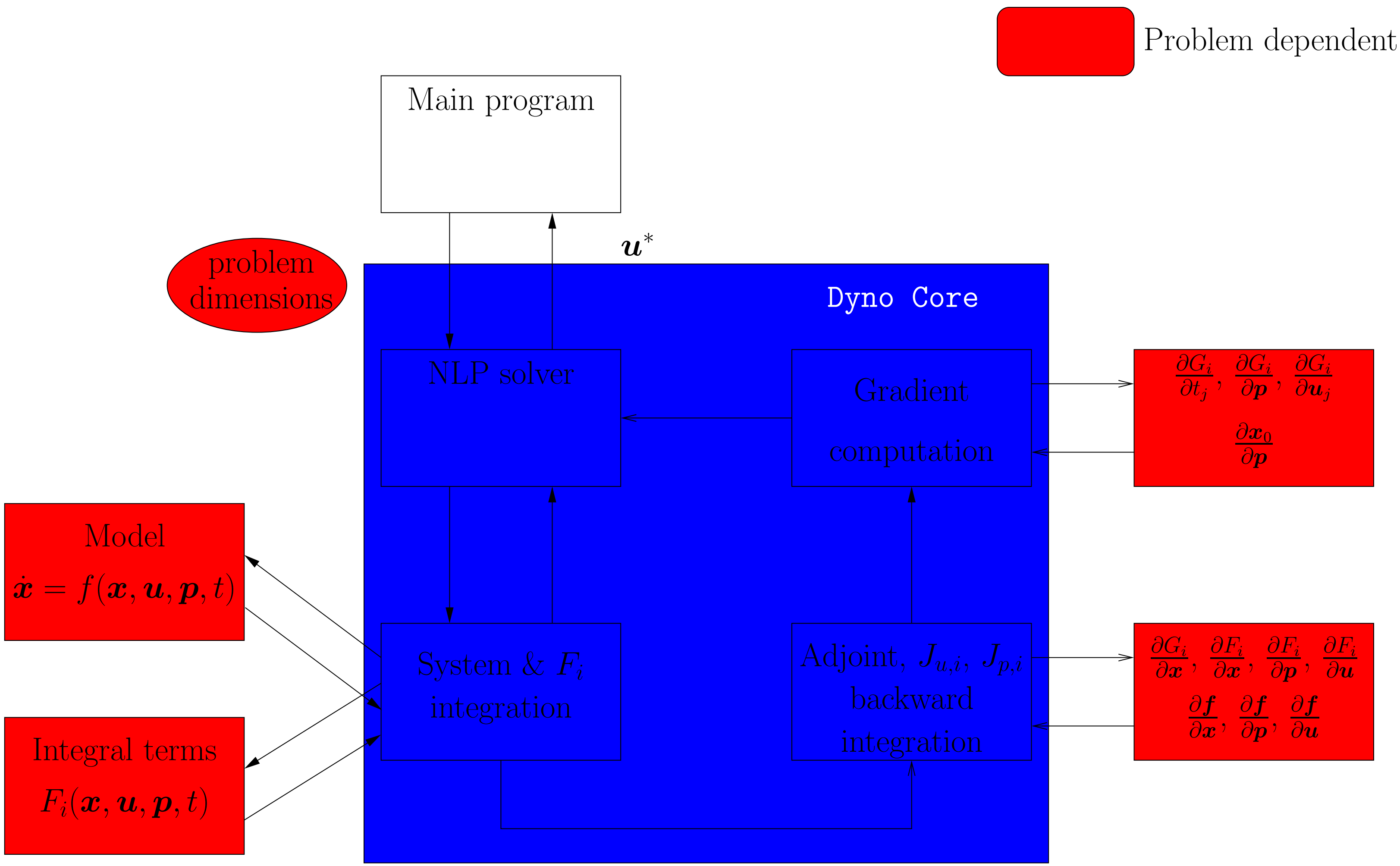


New problem $\rightarrow$ compile new code

Matlab integration



Matlab integration



Dyno Core $\rightarrow$ F77 program $\rightarrow$ MATLAB toolbox (MEX file)
Problem definition $\rightarrow$ MATLAB functions (`.m`)

Matlab integration: why?

- Ease of use: no core modification when changes in
  - Model
  - Cost and constraints
  - Number of time intervals
- Code distribution
  - No need for a Fortran compiler
  - Dyno core source code can be hidden
- Code optimization
  - Dynamic allocation of vectors and matrices
- Performance

☞ Suitable for educational and industrial use

Main drawbacks

- System dependant (works on GNU/Linux with certain g77 /gcc /MATLAB versions)
- Roundoff problems
- ADIFOR not usable $\rightarrow$ no automatic differentiation

Further developments of DYNO

- Use of symbolic computation for derivatives (MATLAB symbolic toolbox?)
- Global optimization

Advertisement ☺

Where to get DYNO?

http://www.kirp.chtf.stuba.sk/~fikar/dyno

Contacts:

miroslav.fikar@stuba.sk

latifi@ensic.inpl-nancy.fr