Numerical Simulation of Dynamic Systems: Hw4 - Solution

Prof. Dr. François E. Cellier
Department of Computer Science
ETH Zurich

March 26, 2013

[H3.15] Backinterpolation With Step-Size Control

We want to repeat Hw.[H3.14] once more, this time using a step-size controlled algorithm. The step-size control to be used is the following. On the *explicit semi-step*, compute now both correctors, and find ε_{rel} according to the formula:

$$\varepsilon_{\text{rel}} = \frac{\|\mathbf{x_1} - \mathbf{x_2}\|_\infty}{\max(\|\mathbf{x_1}\|_2, \|\mathbf{x_2}\|_2, \delta)}$$

If $\varepsilon_{\text{rel}} \leq 10^{-4}$, use the Gustafsson algorithm to compute the step size to be used in the next step:

$$h_{\text{new}} = \left(\frac{0.8 \cdot 10^{-4}}{\varepsilon_{\text{rel}_{\text{now}}}}\right)^{0.06} \cdot \left(\frac{\varepsilon_{\text{rel}_{\text{last}}}}{\varepsilon_{\text{rel}_{\text{now}}}}\right)^{0.08} \cdot h_{\text{old}}$$

except during the first step, when we use:

$$h_{\text{new}} = \left(\frac{0.8 \cdot 10^{-4}}{\varepsilon_{\text{rel}_{\text{now}}}}\right)^{0.2} \cdot h_{\text{old}}$$

[H3.15] Backinterpolation With Step-Size Control II

However, if $\varepsilon_{\text{rel}} > 10^{-4}$, we reject the step at once, i.e., we never even proceed to the implicit semi-step, and compute a new step size in accordance with the same equation as during the first step.

If a step was repeated, the step size for the immediately following next step is also computed according to that equation.

Apply this step-size control algorithm to the same problem as before, and determine the largest global relative error by comparing the solution with the analytical solution of this linear time-invariant system.

[H3.15] Backinterpolation With Step-Size Control III

We start out by implementing the step-size control in the forward semi-step of the $BI4/5_{0.45}$ code:

```
function [xnew, err, tnew, hnew] = bi45tv_step(x, t, h, errl, theta, tol)
    err = 2 * tol;
    rep = 0;
    while err > tol,
        [x_left4, x_left5] = rkf45_step(x, t, theta * h);
        err = norm(x_left4 - x_left5,'inf')/max([norm(x_left4),norm(x_left5), 1.0e - 10]);
        if err > tol,
            h = (0.8 * tol/err) ∧ (0.2) * h;
            rep = 1;
        else
            if errl > 0 & rep == 0,
                hnew = (0.8 * tol/err) ∧ (0.06) * (errl/err) ∧ (0.08) * h;
            else
                hnew = (0.8 * tol/err) ∧ (0.2) * h;
            end,
        end,
    end
    if rep == 1,
        err = -err;
    end
    tnew = t + h;
    xnew = x_left4;
```

[H3.15] Backinterpolation With Step-Size Control IV

```
    err2 = tol;
    H = hessian(x, t, (theta − 1) ∗ h, 5);
    while err2 > 0.1 ∗ tol,
        [dummy, x_right] = rkf45_step(xnew, t + h, (theta − 1) ∗ h);
        xnew = xnew − H\(x_right − x_left4);
        err2 = norm(x_right − x_left4,'inf')/max([norm(x_left4),norm(x_right), 1.0e − 10]);
    end
return
```

The backward semi-step is still the same as before.

[H3.15] Backinterpolation With Step-Size Control V

We also need the analytical solution of this linear system. We compute this in the *frequency domain*, i.e., using *Laplace transform*.
We start by building a system in the time domain:

$S = \mathrm{ss}(A, b, c, d);$

We convert to the frequency domain by computing the *transfer function*:

$G = \mathrm{tf}(S);$

The Laplacian of a step input is $\frac{1}{s}$:

$Pu = 1;$
$Qu = [\,1\ 0\,];$
$U = \mathrm{tf}(Pu, Qu);$

The *input response* in the frequency domain is the product of the transfer function and the input signal:

$Y = G * U;$

[H3.15] Backinterpolation With Step-Size Control VI

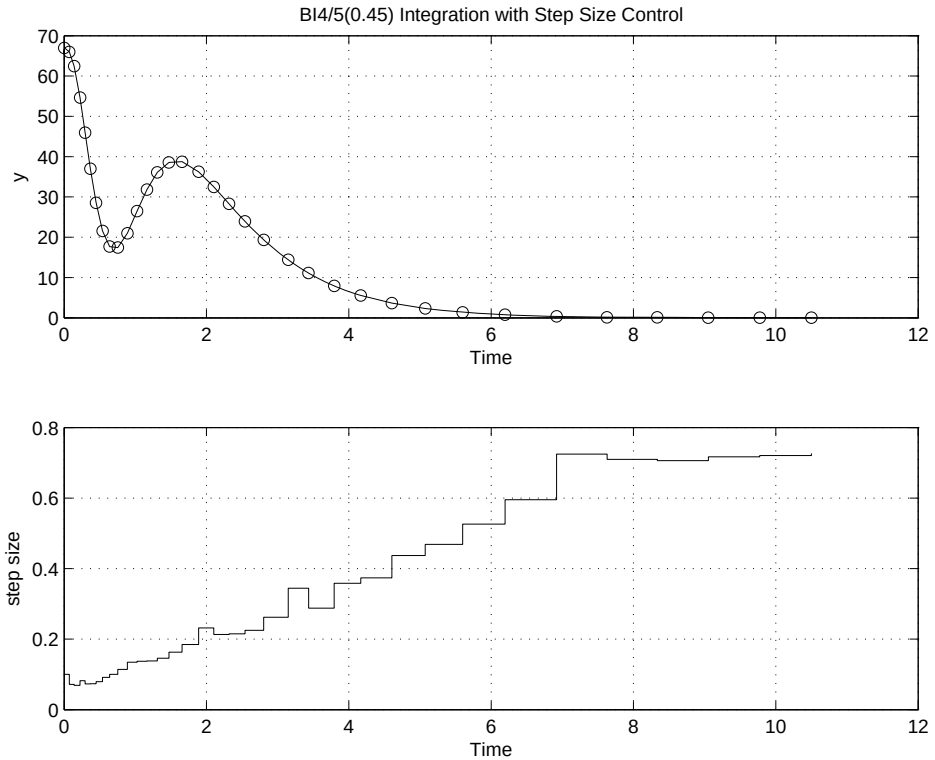
We need to convert the input response back into the time domain. To this end, we perform a *partial fraction expansion*:

```
[Py, Qy] = tfdata(Y,'v');
[r2, l2] = residue(Py, Qy);
```

The analytical solution is the superposition of the *input response* and the *initial state response*:

```
ycorr = zeros(size(yvec));
for i = 1 : nm,
    t = tvec(i);
    y = sum( r2 .* exp(l2 * t) ) + c*expm(A * t) * x0;
    ycorr(i) = y;
end
```

[H3.15] Backinterpolation With Step-Size Control VII



[H3.15] Backinterpolation With Step-Size Control VIII

The average step size used by the algorithm is $h_{avg} = 0.3118$.

The relative error achieved is $err_{rel} = 3.2851e-005$.

This is quite good. We wanted to get $tol = 1.0e-005$, i.e., the actual *global error* is about three times larger than the desired error, but then again, all we do is to control the *local error*. We simply assumed that the local error would be roughly one order of magnitude smaller than the global error (rule of thumb), and therefore, we checked for $err > 0.1 * tol$.

[H3.19] Order Star

Find the *damping order star* for $\mathbf{BI4}/\mathbf{5}_{\mathbf{0.45}}$, and plot it together with the pole and zero locations. Compare with the damping order star of **BI4** that was shown in class.

Find the *frequency order star* for $\mathbf{BI4}/\mathbf{5}_{\mathbf{0.45}}$, and plot it together with the pole and zero locations. Compare with the frequency order star **BI4** that was shown in class.

Finally, compute and plot the *order star accuracy domain* of this method.

For this problem, it may be easier to use MATLAB's contour plot, than your own domain tracking routine.

[H3.19] Order Star II

We need to compute the three-dimensional function $\hat{\sigma}_d(\sigma_d, \omega_d)$. We place a roster of values on the two-dimensional plane spanned by σ_d and ω_d, and compute $\hat{\sigma}_d$ at each roster point:

```
j = sqrt(-1);
svec = zeros(161, 201);
s = [-10 : 0.1 : 10];
w = [-8 : 0.1 : 8];
for i = 1 : 201,
    ss = s(i);
    for k = 1 : 161,
        ww = w(k);
        lambd = ss + j * ww;
        shat = -damp(-lambd, algor);
        svec(k, i) = ss - shat;
    end
end
```

[H3.19] Order Star III

The function damp computes the numerical damping:

```
function shat = damp(a, algor)
    f = ff(-a, 1, algor);
    shat = -log(abs(f));
return
```

whereby the function ff computes the **F**-matrix of the ODE solver:

```
function [F] = ff(A, h, algor)
    I = eye(size(A));
    Ah = A * h;
    theta = 0.45;
    Af = Ah * theta;
    Af2 = Af * Af;  Af3 = Af2 * Af;
    Af4 = Af3 * Af;  Af5 = Af4 * Af;
    Ff = I + Af + Af2/2 + Af3/6 + Af4/24 + Af5/104;
    Ab = Ah * (1 - theta);
    Ab2 = Ab * Ab;  Ab3 = Ab2 * Ab;  Ab4 = Ab3 * Ab;
    Ab5 = Ab4 * Ab;  Ab6 = Ab5 * Ab;
    Fb = I - Ab + Ab2/2 - Ab3/6 + Ab4/24 - Ab5/120 + Ab6/2080;
    F = Fb\Ff;
return
```

based on the $f_4(q)$ and $f_5(q)$ functions that I presented in class for the $RKF4/5$ algorithm.

[H3.19] Order Star IV

We also need to compute the poles and zeros of $f(q)$ for the $BI4/5_{0.45}$ algorithm:

```
function [p, z] = pz(algor)
    z = tf('z', 1);
    theta = 0.45;
    zf = z * theta;
    zf2 = zf * zf;   zf3 = zf2 * zf;
    zf4 = zf3 * zf;   zf5 = zf4 * zf;
    Gf = 1 + zf + zf2/2 + zf3/6 + zf4/24 + zf5/104;
    zb = z * (1 - theta);
    zb2 = zb * zb;   zb3 = zb2 * zb;   zb4 = zb3 * zb;
    zb5 = zb4 * zb;   zb6 = zb5 * zb;
    Gb = 1 - zb + zb2/2 - zb3/6 + zb4/24 - zb5/120 + zb6/2080;
    G = Gb\Gf;
    G = zpk(G);
    [z, p, k] = zpkdata(G,'v');
return
```

[H3.19] Order Star V

We could now cut the three-dimensional function $\hat{\sigma}_d(\sigma_d, \omega_d)$ with a horizontal plane going through the origin. This would give us another way to plot the stability domain of the method. Matlab's contour function will do just that:

contour(σ_d, ω_d, $\hat{\sigma}_d$, [0 0],'k-')

However, we want to get the damping order star, which can be computed in the same fashion:

contour(σ_d, ω_d, ε_σ, [0 0],'k-')

[H3.19] Order Star VI

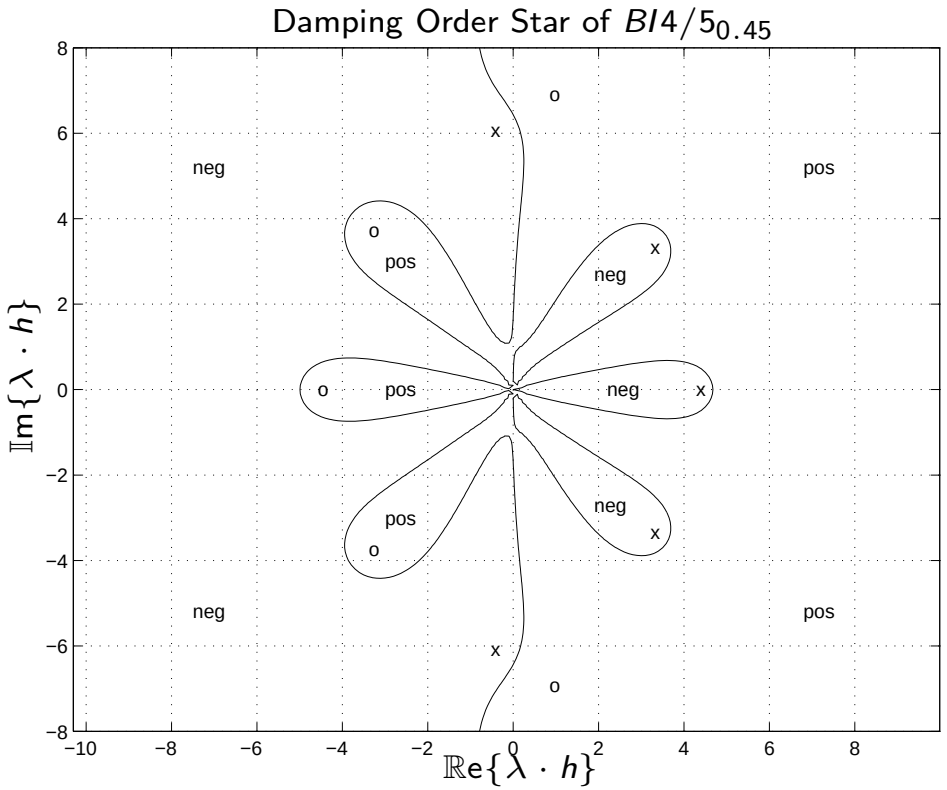


Figure: Damping order star of $BI4/5_{0.45}$ method

[H3.19] Order Star VII

The rational function $f(q)$ has six poles and five zeros, located at:

$$p = \begin{matrix} 21.8413 \\ -0.5210 + 6.0863i \\ -0.5210 - 6.0863i \\ 4.2886 \\ 3.2137 + 3.3423i \\ 3.2137 - 3.3423i \end{matrix}$$

and:

$$z = \begin{matrix} 0.8496 + 6.9172i \\ 0.8496 - 6.9172i \\ -4.5705 \\ -3.3792 + 3.7377i \\ -3.3792 - 3.7377i \end{matrix}$$

[H3.19] Order Star VIII

The frequency order star can be computed in the same fashion. We now need to compute the three-dimensional function $\hat{\omega}_d(\sigma_d, \omega_d)$. To this end, we need a function freq:

```
function ohat = freq(a, algor)
    f = ff(a, 1, algor);
    ohat = atan2(imag(f),real(f));
return
```

[H3.19] Order Star IX

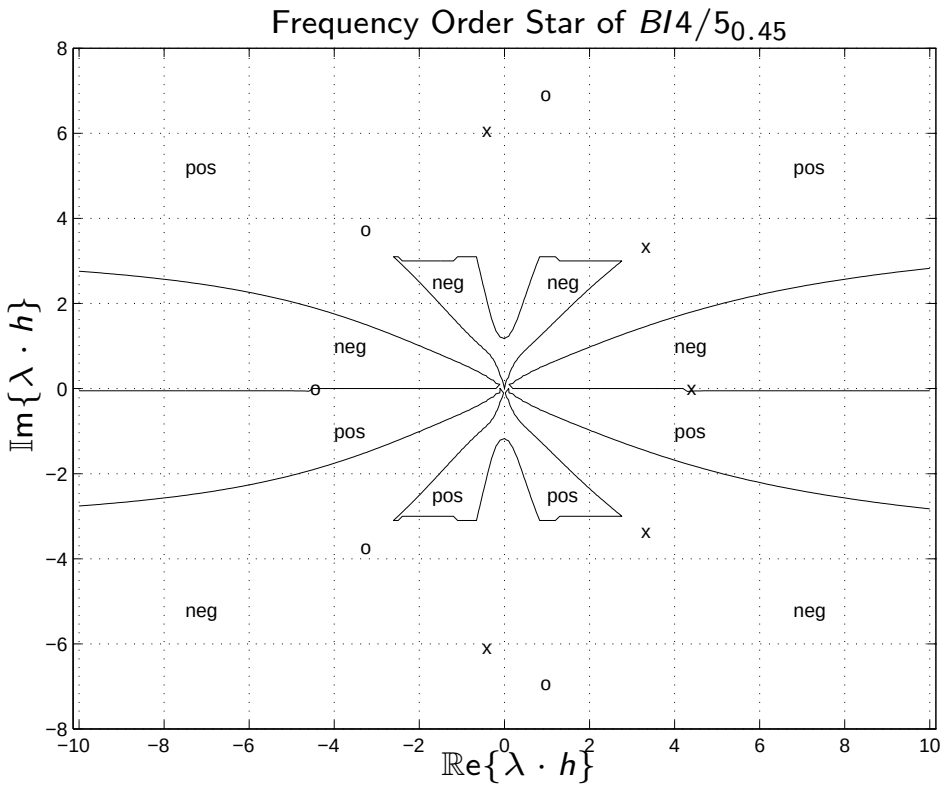


Figure: Frequency order star of $BI4/5_{0.45}$ method

[H3.19] Order Star X

The order star accuracy domain is once again computed in the same fashion. Now, we compute os_{err} at each roster point:

```
shat = -damp(-lambd, algor);
ohat = atan2(imag(f),real(f));
errvec(k, i) = abs(ss - shat) + abs(ww - ohat);
```

and plot:

```
tol = 1.0e - 4;
contour(s, w, errvec, [tol tol],'k-')
hold on
tol = 1.0e - 3;
contour(s, w, errvec, [tol tol],'k-')
tol = 1.0e - 2;
contour(s, w, errvec, [tol tol],'k-')
```

[H3.19] Order Star XI

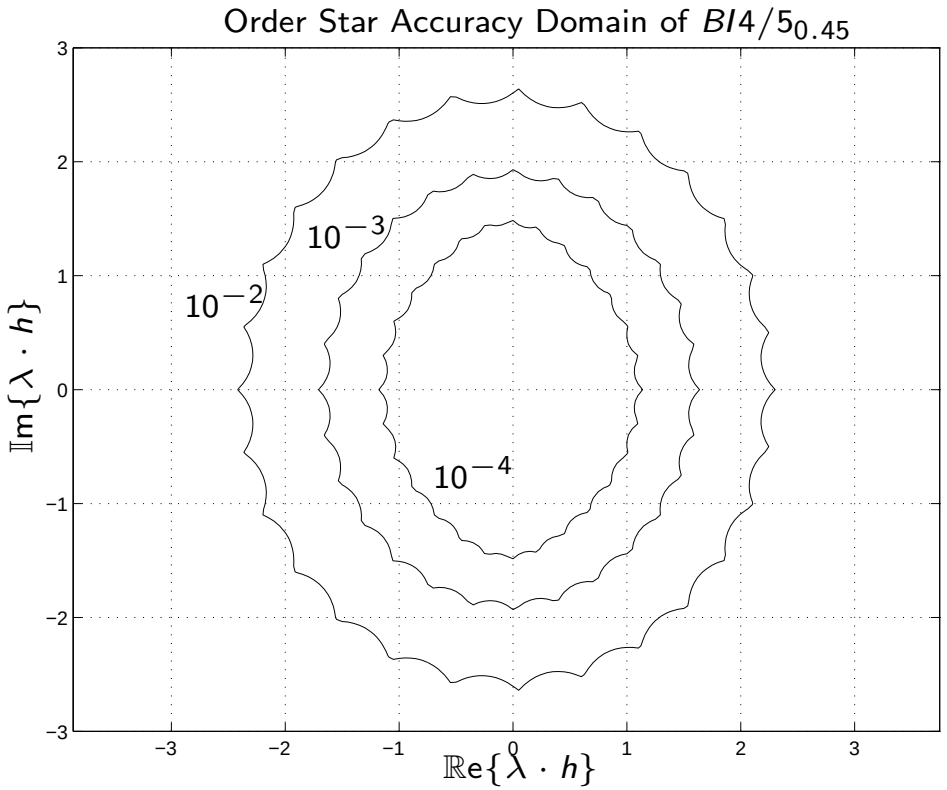


Figure: Order star accuracy domain of $BI4/5_{0.45}$ method