I. REDEFINING HOW IMAGES ARE DISPLAYED

[frame=lines,fontsize=,linenos]common-lisp (defun org-display-inline-images (optional include-linked refresh beg end) "Display inline images. Normally only links without a description part are inlined, because this is how it will work for export. When INCLUDE-LINKED is set, also links with a description part will be inlined. This can be nice for a quick look at those images, but it does not reflect what exported files will look like. When REFRESH is set, refresh existing images between BEG and END. This will create new image displays only if necessary. BEG and END default to the buffer boundaries." (interactive "P") (when (display-graphic-p) (unless refresh (org-remove-inline-images) (if (fboundp 'clear-image-cache) (clear-image-cache))) (save-excursion (save-restriction (widen) (setq beg (or beg (point-min)) end (or end (point-max))) (goto-char beg) (let ((re (concat "

)

)

([[]+?"(substring(org − image − file − name − regexp)0 − 2)"

)

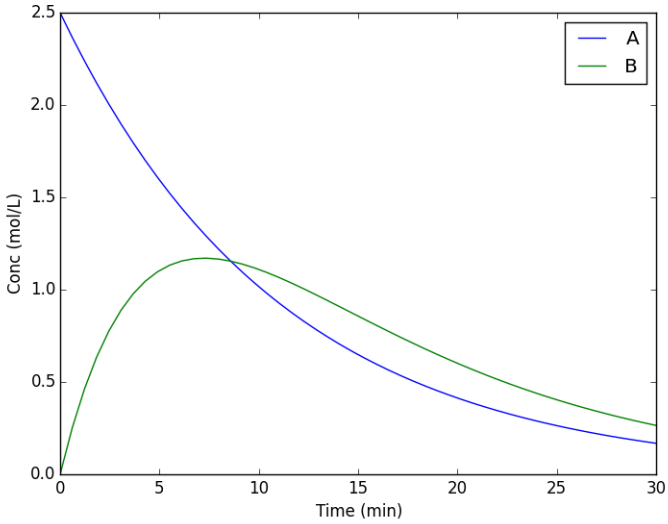
]"(ifinclude − linked""""

]")))(case − fold − searcht)oldfileovimgtypeattrwidthwidth)(while(re − search − forwardreendt)(setqold(get − char − property − and − overlay(match − beginning1)′org − image − overlay)file(expand − file − name(concat(or(match − string3)"")(match − string4))))(when(image − type − available − p′imagemagick)(setqattrwidth(if(or(listporg − image − actual−width)(nullorg−image−actual−width))(save−excursion(save−match−data(when(re−search−backward;;Imodifiedthisregexp" + attr.∗ : width[¨] + ([0 − 9] +)(in|px|cm)"(save − excursion(re − search − backward"[¨]∗

—

'" nil t)) t) ;; and here, if we catch units, we fall back on org-image-actual-width ;; by setting attrwidth to nil (if (match-string 2) nil (string-to-number (match-string 1))))))) width (cond ((eq org-image-actual-width t) nil) ((null org-image-actual-width) attrwidth) ((numberp org-image-actual-width) org-image-actual-width) ((listp org-image-actual-width) (or attrwidth (car org-image-actual-width)))) type (if width 'imagemagick))) (when (file-exists-p file) (if (and (car-safe old) refresh) (image-refresh (overlay-get (cdr old) 'display)) (setq img (save-match-data (create-image file type nil :width width))) (when img (setq ov (make-overlay (match-beginning 0) (match-end 0))) (overlay-put ov 'display img) (overlay-put ov 'face 'default) (overlay-put ov 'org-image-overlay t) (overlay-put ov 'modification-hooks (list 'org-display-inline-remove-overlay)) (push ov org-inline-image-overlays)))))))))

`org-display-inline-images`



`org-display-inline-images`